RTO-MP-048
AC/323(IST)TP/7

RTO-MP-048

NORTH ATLANTIC TREATY ORGANIZATION

RESEARCH AND TECHNOLOGY ORGANIZATION

BP 25, 7 RUE ANCELLE, F-92201 NEUILLY-SUR-SEINE CEDEX, FRANCE

RTO MEETING PROCEEDINGS 48

Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"

(l'Utilisation des produits vendus sur étagères dans les applications militaires de défense "l'Exploitation sans merci des produits commerciaux")

Papers presented at the Information Systems Technology Panel (IST) Symposium held in Brussels, Belgium, 3-5 April 2000.

Published December 2000

Distribution and Availability on Back Cover

Form SF298 Citation Data

Report Date *("DD MON YYYY")* 01122000	**Report Type** N/A	**Dates Covered (from... to)** *("DD MON YYYY")*

Title and Subtitle Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"	**Contract or Grant Number**
	Program Element Number
Authors	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) Research and Technology Organization North Atlantic Treaty Organization BP25, 7 rue Ancelle, F-92201 Neuilly-sur-Seine, Cedex, France	**Performing Organization Number(s)**
Sponsoring/Monitoring Agency Name(s) and Address(es)	**Monitoring Agency Acronym**
	Monitoring Agency Report Number(s)
Distribution/Availability Statement Approved for public release, distribution unlimited	
Supplementary Notes	
Abstract	
Subject Terms	
Document Classification unclassified	**Classification of SF298** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** unlimited
Number of Pages 214	

RTO-MP-048
AC/323(IST)TP/7

NORTH ATLANTIC TREATY ORGANIZATION

RESEARCH AND TECHNOLOGY ORGANIZATION

BP 25, 7 RUE ANCELLE, F-92201 NEUILLY-SUR-SEINE CEDEX, FRANCE

RTO MEETING PROCEEDINGS 48

Commercial Off-the-Shelf Products in Defence Applications “The Ruthless Pursuit of COTS”

(l’Utilisation des produits vendus sur étagères dans les applications militaires de défense “l’Exploitation sans merci des produits commerciaux”)

Papers presented at the Information Systems Technology Panel (IST) Symposium held in Brussels, Belgium, 3-5 April 2000.

The Research and Technology Organization (RTO) of NATO

RTO is the single focus in NATO for Defence Research and Technology activities. Its mission is to conduct and promote cooperative research and information exchange. The objective is to support the development and effective use of national defence research and technology and to meet the military needs of the Alliance, to maintain a technological lead, and to provide advice to NATO and national decision makers. The RTO performs its mission with the support of an extensive network of national experts. It also ensures effective coordination with other NATO bodies involved in R&T activities.

RTO reports both to the Military Committee of NATO and to the Conference of National Armament Directors. It comprises a Research and Technology Board (RTB) as the highest level of national representation and the Research and Technology Agency (RTA), a dedicated staff with its headquarters in Neuilly, near Paris, France. In order to facilitate contacts with the military users and other NATO activities, a small part of the RTA staff is located in NATO Headquarters in Brussels. The Brussels staff also coordinates RTO's cooperation with nations in Middle and Eastern Europe, to which RTO attaches particular importance especially as working together in the field of research is one of the more promising areas of initial cooperation.

The total spectrum of R&T activities is covered by 7 Panels, dealing with:

• SAS	Studies, Analysis and Simulation
• SCI	Systems Concepts and Integration
• SET	Sensors and Electronics Technology
• IST	Information Systems Technology
• AVT	Applied Vehicle Technology
• HFM	Human Factors and Medicine
• MSG	Modelling and Simulation

These Panels are made up of national representatives as well as generally recognised 'world class' scientists. The Panels also provide a communication link to military users and other NATO bodies. RTO's scientific and technological work is carried out by Technical Teams, created for specific activities and with a specific duration. Such Technical Teams can organise workshops, symposia, field trials, lecture series and training courses. An important function of these Technical Teams is to ensure the continuity of the expert networks.

RTO builds upon earlier cooperation in defence research and technology as set-up under the Advisory Group for Aerospace Research and Development (AGARD) and the Defence Research Group (DRG). AGARD and the DRG share common roots in that they were both established at the initiative of Dr Theodore von Kármán, a leading aerospace scientist, who early on recognised the importance of scientific support for the Allied Armed Forces. RTO is capitalising on these common roots in order to provide the Alliance and the NATO nations with a strong scientific and technological basis that will guarantee a solid base for the future.

The content of this publication has been reproduced
directly from material supplied by RTO or the authors.

Published December 2000

ISBN 92-837-1049-5

ST JOSEPH
OTTAWA/HULL

Printed by St. Joseph Ottawa/Hull
(A St. Joseph Corporation Company)
45 Sacré-Cœur Blvd., Hull (Québec), Canada J8X 1C6

Commercial Off-the-Shelf Products in Defence Applications “The Ruthless Pursuit of COTS”

(RTO MP-048)

Executive Summary

Commercial Off The Shelf (COTS) software packages have been proposed for many military applications, including embedded systems, communication systems, operating systems, and in some cases critical military applications. A primary reason for proposing COTS for military applications is an assumption that software lifecycle costs would be substantially reduced. Such a mandate has major implications for acquisition, design, production, evaluation, and testing of systems that must maintain high levels of assurance.

Verified and validated levels of software quality, safety, reliability, sustainability, and survivability can be difficult to obtain and are often expensive to achieve. However, critical military applications demand levels of software assurance that most vendors do not apply to their commercial products. Although the cost saving benefits of COTS packages for non-critical applications is undisputed, there continues to be an on-going debate on the cost benefits of COTS software for critical applications (military or commercial).

Various approaches have been proposed for COTS utilization for military systems. One approach is to adopt COTS software for non-critical military applications, where an organization’s operations concept is modified to be consistent with the commerciality properties of a COTS software package. A second approach is to adapt COTS software for military applications, where the original source is modified to be consistent with unique operational requirements. A third approach is to modify COTS software for critical military applications, where an independent testing organization obtains the original vendor source code for assurance testing. Modified COTS software generally requires a substantial change to vendor source code. A fourth approach assumes that COTS software packages cannot be adequately evaluated or verified, and should not be used for any critical military systems.

In order to address procurement, design, evaluation, testing, verification, validation, adoption, adaptation, and modification issues associated with the acquisition and utilization of COTS software packages for military systems, NATO hosted a three-day symposium in Brussels, Belgium. The symposium consisted of two keynote speakers, and six technical sessions consisting of twenty-four presentations.

The symposium treated the subject with rigor that is characteristic of a mature engineering discipline. This symposium and its products will be a standard by which COTS software evaluation and certification is measured for years to come. Presentations were thorough, accurate, and current. Several presentations actually anticipated results that have yet to appear in archival journals.

l’Utilisation des produits vendus sur étagères dans les applications militaires de défense “l’Exploitation sans merci des produits commerciaux”

(RTO MP-048)

Synthèse

Des progiciels disponibles sur étagère (COTS) ont été proposés pour de nombreuses applications militaires, y compris pour des logiciels intégrés, des systèmes de communication, des systèmes d’exploitation et, dans certains cas, des applications militaires indispensables à la mission. L’une des principales raisons pour laquelle le matériel COTS est proposé pour des applications militaires réside dans le fait que son achat permettrait de réduire considérablement les coûts globaux de possession des logiciels. Une telle proposition a des conséquences majeures pour l’acquisition, la conception, la fabrication, l’évaluation et les essais de systèmes censés garantir des hauts niveaux de fiabilité militaires.

Des niveaux vérifiés et validés de qualité, de sécurité, de fiabilité, de soutenabilité et de survivabilité des logiciels peuvent être difficiles et coûteux à obtenir. Cependant, les logiciels demandés pour les applications militaires indispensables à la mission exigent des niveaux de fiabilité qui sont rarement rencontrés dans le commerce. Bien que les économies de coûts résultant des achats COTS pour des applications non-décisives soient indiscutables, le débat sur les coûts-avantages des logiciels COTS pour des applications indispensables à la mission (militaires ou commerciales), reste d’actualité.

Différentes approches ont été proposées pour la mise en œuvre de produits COTS dans les systèmes militaires. L’une d’entre elles consiste à adopter les logiciels COTS pour des applications militaires non-critiques, où le concept d’opérations d’une organisation est modifié pour être compatible avec les caractéristiques commerciales d’un progiciel COTS. Une deuxième approche consiste à adapter les logiciels COTS aux applications militaires, où la source est modifiée pour la rendre compatible avec des besoins opérationnels spécifiques. Une troisième approche consiste à modifier des logiciels COTS destinés à des applications militaires, où une organisation d’essais indépendante obtient le code source du fournisseur afin de réaliser des essais de fiabilité. En général, la modification des logiciels COTS entraîne, à son tour, des modifications considérables au niveau du code source du fournisseur. Dans une quatrième approche, il est supposé que les progiciels COTS ne peuvent être ni évalués ni vérifiés de façon satisfaisante, et que par conséquent, ils ne doivent pas être utilisés dans des systèmes militaires indispensables à la mission.

L’OTAN a organisé un symposium de trois jours à Bruxelles en Belgique, afin d’examiner l’approvisionnement, la conception, l’évaluation, les essais, la validation, l’adoption, l’adaptation et la modification dans le cadre de l’acquisition et la mise en œuvre de progiciels COTS pour applications militaires. Le programme du symposium a comporté deux discours d’ouverture et six sessions techniques, qui ont permis la présentation de vingt-quatre communications.

Le sujet a été traité avec la rigueur caractéristique d’une discipline d’ingénierie déjà au point. Ce symposium et les documents associés vont représenter une norme pour l’évaluation et la certification des logiciels COTS pendant de nombreuses années. Les présentations étaient complètes, précises et d’actualité. Dans certains cas, elles ont même fait état de résultats qui ne sont pas encore apparus dans la presse spécialisée.

Click inside the blue boxes or on the titles to view the corresponding section

Contents

	Page
Executive Summary	**iii**
Synthèse	**iv**
Theme/Thème	**vii**
Information Systems Technology Panel	**viii**

	Reference
Technical Evaluation Report by M.S. Fisher	**T**
Robust Nonproprietary Software by P.G. Neumann	**KN1**

SESSION I: ACADEMIC PERSPECTIVE:
COTS ACQUISITION, UTILISATION AND EVALUATION
Chairman: Dr M. SLOAN (US)

Wrapping the COTS Dilemma by I. White	**1**
The COTS IT Circle by A. Weiss	**2**
Standards – Myths, Delusions and Opportunities by N. Peeling and R. Taylor	**3**

SESSION II: COTS ACQUISITION CHALLENGES
Chairman: Dr M. VIGDER (CA)

***E*nvironment for *S*ignal *P*rocessing *A*pplication *D*evelopment and Pr*O*totypi*N*g - ESPADON** by B. Madahar, J. Hunink, G. Edelin, J. Smith and B. Saget	**4**
United States Army Commercial Off-the-Shelf (COTS) Experience: The Promises and Realities by J.J. Barbarello and W. Kasian	**5**
The Coordinated Defence Role in Civil (Telecom) Standardisation by J.P. Thorlby	**6**
Risks by Using COTS Products and Commercial ICT Services by S. Jantsch	**7**
C3I Systems Acquisition and Maintenance in Relation to the Use of COTS Products by S. Rampino and M. Fiorilli	**8**
COTS Software Evaluation Techniques by J.C. Dean and M.R. Vigder	**9**

SESSION III: COTS: EVALUATION AND ASSURANCE
Chairman: Dr I. WHITE (UK)

Reliable Tailored-COTS via Independent Verification and Validation 10
by M.A. Beims and J.B. Dabney

COTS Software Supplier Identification and Evaluation 11
by A. Miller

Maintaining COTS-Based Systems 12
by M.R. Vigder and J. Dean

Detection of Malicious Code in COTS Software via Certifying Compilers 13
by R. Charpentier and M. Salois

Application of COTS Communications Services for Command and Control of Military Forces 14
by P. Kerr and J. McCarthy

Confidently Integrating COTS Software Under Worst Case Assumptions KN2
by J. Voas

The Convergence of Military and Civil Approaches to Information Security? 15
by R. Rowlingson

Dynamic Detection of Malicious Code in COTS Software 16
by M. Salois and R. Charpentier

The Ruthless Pursuit of the Truth about COTS 17
by N.F. Schneidewind

Determining the Suitability of COTS for Mission Critical Applications 18
by R.J. Kohl

SESSION IV: VENDOR PERSPECTIVE: COTS
Chairman: Major W. TACK (BE)

Six Facets of the Open COTS Box 19
by D.H. Dumas

Lotus White Paper on COTS Software for Military Crisis Applications 20
by P. Fournery and U. Sorensen

Wireless TCP/IP and Combination with Broadband Media 21
by T.A. Kneidel

COTS Based Systems: The Necessity of a Service & Systems Management Strategy to Assure Service Levels 22
by D. Somerling

SESSION V: USER PERSPECTIVE: COTS
Chairman: Major W. TACK (BE)

Modernizing OMIS, an Operational Airforce C2 System, Using COTS Hardware and Software Products 23
by J.G. Stil

SESSION VI: COTS: INTEGRATION
Chairman: Major W. TACK (BE)

Experiences in Designing Radio Monitoring Systems Using Commercial Off-the-Shelf (COTS) Components 24
by G. Palten

Theme

Industrial and commercial-grade information technology (IT) products such as workstations, networking products, and databases have long been employed by the military. While these are clearly commercial-off-the-shelf (COTS) information technology products, the term COTS now commonly includes commodity personal computers, operating systems and productivity tools designed for the consumer market. Commercial office automation suites, electronic mail, databases, and similar business-oriented software are often directly applicable to military needs and can be run effectively on inexpensive personal computers. The appropriate use of personal computers, networks and off-the-shelf software products for military applications is an effective way to improve efficiency while coping with limited budgets and reduced staff. Recently it has been proposed that these same products be employed as mandated platforms, operating systems and major software elements for all but the most specialised defence applications. Such a mandate has major implications for design, production and employment of systems able to maintain military levels of assurance. This symposium will address NATO interests and issues in employing COTS hardware and software while maintaining required levels of system assurance.

TOPICS TO BE COVERED:

1) Standards and standardisation
2) Consumer, commercial, and industrial COTS availability and assurance properties
3) Methods for providing high assurance while employing low assurance products
4) Interoperability and software product migration
5) Obsolescence and upgrade policy
6) Integration with legacy systems
7) Interoperability with coalition systems

Thème

Des produits informatiques (IT) industriels et du commerce tels que postes de travail, produits conçus pour le travail en réseau et bases de données, sont en service dans les armées depuis longtemps. Bien qu'il s'agisse évidemment de produits informatiques du commerce (COTS), le terme COTS s'utilise aussi aujourd'hui pour les ordinateurs personnels, les systèmes d'exploitation et les outils de productivité destinés au grand public. Les programmes de bureautique, le courrier électronique, les bases de données et autres logiciels de bureau sont, en effet, souvent utilisables directement pour les tâches militaires et peuvent être exploités de façon satisfaisante sur des ordinateurs personnels de coût modique. La mise en œuvre judicieuse d'ordinateurs personnels, de réseaux et de logiciels du commerce pour des applications militaires devrait ainsi permettre de travailler plus efficacement dans un contexte de réduction d'effectifs et de restrictions budgétaires. Il a été proposé récemment d'utiliser ces produits comme plates-formes, systèmes d'exploitation et progiciels autorisés pour toutes les applications militaires, à l'exception des plus sensibles. Une telle orientation a des conséquences majeures pour la conception, la fabrication et la mise en œuvre de systèmes devant garantir des niveaux de sécurité compatibles avec les missions. Ce symposium examinera à la fois les avantages possibles pour l'OTAN et la compatibilité entre la mise en œuvre de matériels et de logiciels COTS et le maintien des niveaux de sécurité des systèmes requis par ces missions.

SUJETS A TRAITER :

1) Normes et normalisation
2) Disponibilité et fiabilité des produits COTS grand public, commerciaux et industriels
3) Méthodes susceptibles d'assurer un haut niveau de sécurité de fonctionnement avec des produits de niveau élémentaire
4) Interopérabilité et migration des logiciels
5) Obsolescence et politiques de modernisation
6) Intégration dans des systèmes existants
7) Interopérabilité entre systèmes au sein d'une coalition.

Information Systems Technology Panel

Chairman

Dr M. VANT
Deputy Director General
Defence Research Establishment Ottawa
Dept of National Defence
3701 Carling Ave
OTTAWA, ONTARIO, K1A 0K2, CANADA

Deputy Chairman

Dr R. JACQUART
Directeur du DTIM
ONERA/DTIM
BP 4025
31055 TOULOUSE CEDEX 4, FRANCE

TECHNICAL PROGRAMME COMMITTEE

Chairman:	Dr L BLAZY	US
Members:	Maj W TACK	BE
	Dr G VEZINA	CA
	Dr I WHITE	UK
	Prof M SLOAN	US

PANEL EXECUTIVE

From Europe:

RTA-OTAN
Lt-Col A GOUAY, FAF
IST Executive
BP 25, 7 rue Ancelle
F-92201 NEUILLY SUR SEINE CEDEX, FRANCE

Telephone: 33-1-5561 2280/82 - Telefax: 33-1-5561 2298/99

From the USA or CANADA:

RTA-NATO
Attention: IST Executive
PSC 116
APO AE 09777

HOST NATION LOCAL COORDINATOR

Major W TACK
General Staff, JSM-R&T/Space
Everestraat, 1
B-1140 BRUSSELS, BELGIUM
Tel: (32) 2 701 66 15
FAX: (32) 2 701 66 20

ACKNOWLEDGEMENTS/REMERCIEMENTS

The IST Panel wishes to express its thanks to the RTB members from Belgium for the invitation to hold this Symposium in Brussels and for the facilities and personnel which made the Symposium possible.

Les membres de la commission IST tiennent à remercier les membres du RTB de la Belgique pour leur invitation à tenir cette réunion à Bruxelles, ainsi que pour les installations et le personnel mis à sa disposition.

Technical Evaluation Report

Marcus S. Fisher
NASA Ames IV&V Facility
100 University Dr.
Fairmont, WV 26554, USA

Abstract

It has been proposed that Commercial Off-The-Shelf (COTS) products be procured for applications within military systems. Such a proposal can generate significant concern when COTS products are used for mission-critical systems. Having a significant interest in this approach, NATO organized a three-day symposium to address risks, benefits, and issues associated with COTS acquisition, utilization, and assurance. The purpose of this paper is to evaluate overall technical merit of this symposium.

Introduction

In an attempt to advance "the engineering" of software, organizations frequently adopt new technologies that promise to decrease software costs and improve the delivery time. Armed with such promises many organizations believed they had discovered a Holy Grail for software development. By what measures do we conclude that integrating such promising technologies, such as COTS, do in fact live up to there promised benefits?

The use of Commercial-Off-The-Shelf (COTS) products is one of these promising technologies. Recently it has been proposed that COTS products be employed as mandated platforms, operating systems, and major software components for military applications. Such a mandate has major implications for acquisition, design, production, test, and deployment of such systems that must maintain high levels of assurance. The use of COTS components for military applications proposes to increase the efficiency of development and deployment while maintaining and even decreasing the cost of such systems.

NATO has a significant interest in resolving these issues and organized a three-day symposium to address benefits, risks, and issues involved when utilizing COTS components in critical military applications. They have strategically identified key topics with the intent of forging a common understanding of the practice, problems, and possible paths to take when engineering COTS based systems. They acquired multiple viewpoints from many NATO countries that addressed the following topics:

- Standards and Standardization,
- COTS availability and assurance properties,
- Methods for providing high assurance while employing low assurance products,
- Interoperability and software product migration,
- Obsolescence and upgrade policy,
- Integration with legacy systems, and
- Interoperability with coalition systems.

Evaluation

The three-day symposium consisted of two keynote speakers and six technical sessions:

- Session I: Academic Perspective: COTS Acquisition, Utilization, and Evaluation
- Session II: COTS Acquisition Challenges
- Session III: COTS Evaluation and Assurance
- Session IV: Vendor Perspective
- Session V: User Perspective
- Session VI: COTS Integration

This section addresses each session individually, identifies emerging technologies presented, and impacts these technologies can have on successful COTS integration.

Two keynote presentations established a foundation for the many presentations that followed. Attendees were reminded of the desirable attributes of critical information systems, varying definitions of COTS components, problems/risks associated with using COTS, open-source versus closed-source software, as well as an innovative approach for certifying and guaranteeing the correctness of COTS software components.

In order to meet assurance levels of defense applications, COTS developers must provide guarantees about their software behavior, which should be captured in a Software Quality Warranty. Several approaches were presented:

1) Process Certification - incorporating and establishing standard practices for developing software (e.g., ISO, CMM, IEEE)
2) Personnel Certification - certifying the developers themselves (e.g., IEEE)
3) Product Certification - The end goal is to create a product based quality evaluation method

The approaches presented above may increase the quality of software being developed, but only one assures the product itself is of high quality. Process certification implies a certified software development process produces high quality software. Personnel certification implies that certified personnel always produce quality software. Product certification provides a methodology for assuring COTS products.

A product certification approach presents significant controversy because it would require a major paradigm change in industry. However, it does seem logical that one start advocating the production of quality software. What is missing? A paradigm to quantitatively assess software quality based on rigorous product testing techniques with very well defined environmental assumptions.

A software certification approach requires collaborative efforts between military and industry personnel to define a suitable paradigm for developers of COTS components to follow, and to certify their software products.

Session I

Session one focused on an academic discussion of issues associated with COTS acquisition, utilization, and evaluation. Basic assumptions included that using COTS components in software systems is inevitable, people perceive that software components are cheaper to buy than build, and that software products are buggy.

There were several problems introduced that can be remedied by science and engineering practices. One such problem is that there does not exist a formal system's assurance paradigm for COTS based systems. The questions that need to be answered include the following:

1) Is the product properly specified?
2) What guarantees are given of its performance?
3) What is its reliability in performing the task?
4) What undeclared features are present (a.k.a. Easter Eggs)?

Although the need for an assurance paradigm was emphasized, current methodologies do not address the assurance aspects from a system's perspective.

Methodologies such as black box testing, conformance testing, use of safety critical techniques, and code analysis were presented. However, they do not take a systems approach, they do not address integration issues, nor do they provide for a system perspective.

A few innovative techniques for product assurance were introduced that do warrant further consideration:

- Wrapping the COTS components, and
- Developing formal behavior specifications.

Wrapping COTS component takes on the perspective that we can't cure the inherent COTS faults, so we must live with them. By developing code and inserting it between a component and a system, we would insulate the system from erroneous behavior. We have seen this approach effectively employed for legacy code and even in newer technologies such as Common Object Request Broker Architecture (CORBA), where we define the interfaces components support. This approach is unique because now we are employing a methodology to guard against non-specified behaviors, which act as a filter to protect the system as a whole.

Behavioral specifications formally specify a component's behavior, which encapsulates the completeness criteria for black box specification requirements. However there are a few drawbacks, such as the non-trivial costs associated to produce a detailed specification, and COTS suppliers must be willing to develop them. An advantage for developing such a specification is that all customers can theoretically share the same specification, which gives the producers a market advantage over their competitors.

In addition to a lack of a formal systems assurance paradigm, other problems were noted that surfaced during the symposium such as:

- The lack of a "Best Practices" forum, and a user group focused entirely on COTS based systems', and
- The effects COTS have on configuration management.

One concluding observation was the lack of a centralized forum for disseminating best practices, lessons learned, successes, failures, standards and recommendations, et cetera. Techniques that have paid off in the past are often overlooked because of lack of broad participation.

Configuration management practices need to address change when dealing with components that are constantly being upgraded. As newer versions for components surface, the upgrade to the newer version requires considerable attention. Attention that current techniques do not address.

Session II

Session two focused on acquisition challenges associated with COTS based systems. Customers are experiencing a major paradigm shift in the software development process. Not only has the military mandated the use of COTS products, current approaches used to develop software either lend themselves easily to the use of COTS, (e.g., prototyping or Rapid Application Development (RAD)), or are not at all, (e.g., waterfall). This has motivated the genesis of newer development methodologies.

One innovative approach to COTS acquisition is to support software reuse, concurrent engineering, and rapid prototyping. Other presentations showed through a series of case studies when to "Adopt" a COTS package, when to "Adapt" a COTS package into the system, and when to "Develop" software components.

Additional discussions focused on where and when do you evaluate and select a COTS package? There have been attempts to provide COTS evaluation techniques but they rely on traditional development paradigms and highly structured requirements. An alternative methodology discussed was one in which the COTS software selection is done in parallel with the requirements definition. Using this approach allows the system requirements to be massaged to enable easy adaptation of predefined COTS components. This does present controversy because this technique suggests the creation and evolution of system requirements are influenced and even changed by the availability of COTS products. Do we allow our system expectations to be determined by the functionality and availability of COTS products?

Another major issue discussed was where do standards fit into to the COTS acquisition process? Standards have shown to provide interoperability, market development, competition, and they leverage on past experiences.

We have touched on facets of software development life cycles, incorporating standards, and assuming a systems perspective. What is lacking are risk management or analysis techniques required as a result of these new development and COTS component scenarios. We have not explored whether our current methodologies of risk management can be incorporated into these innovative techniques or if they need to be evolved.

Session III

Session three addressed evaluation and assurance techniques currently being employed for COTS intensive systems. Discussions addressed why we should even consider investing into this technology and concluded that since the military has become a key player and major customer of COTS products then they need to be a major driver in establishing requirements for COTS developers.

Taking COTS component that works well in one environment, and then adapting it into another environment is not always effective. We are faced with the following challenges when integrating COTS:

- Meeting all the requirements and nothing more,
- High reliability from the product,
- Constant availability,
- High quality from the product, and
- Rigorous recovery requirements.

To meet these challenges some proposed techniques include:

- Robust verification plans,
- Early prototyping,
- Good relations with the COTS producers,
- Up front systems engineering evaluations, and
- Insight into the product.

In addition maintenance plans need to be more robust of the product changes ensued, which indirectly effects the entire life cycle, especially the Operations and Maintenance (O&M) phases. It was shown that for COTS intensive systems a large percentage of the development costs fall into the maintenance phase. The maintenance of such systems has many issues:

- Focusing on high level COTS products and not low level source code,
- Evolution of COTS products is under the control of the product developer,
- Visibility into the product is limited, and
- Tailoring and gluing code together is intensive.

Although it is advantageous to evaluate the COTS product, some others recommended we evaluate the producers of COTS products. By producing a repository of carefully and rigorously evaluated developers of COTS products a more efficient environment for selecting COTS packages can be provided. There were two approaches proposed:

1) Software Process Assessment
2) Tailored CMM Assessment

The symposium also experienced dynamic integration and analysis techniques. Techniques to consider include:

- "Intelligent Agents" that roam a system, profiling the execution and avoiding any erroneous behaviors,
- Signature or heuristic based analysis,
- Fault injection and wrapping, and
- Certification while compiling.

Certify while compiling was presented as a means to perform formal verification on the intended COTS packages. By creating a formal specification of the expected behavior a certification compiler can prove that the specification holds throughout the COTS package. This provides a sound scientific and mathematical approach to prove the component behaves as intended, however the feasibility of this technique was not addressed. Can entry-level development teams incorporate such a technique? Of course we do not administer this technique on every Source Line Of Code (SLOC), so it may be feasible to employ this technique on identified critical vectors traced through the code. We take a similar approach when we fold the state space during model checking.

Adapting a COTS product to an operational environment, for which it was not intended, may achieve a cost benefit and still achieve system reliability requirements when augmented with an appropriate Independent Verification & Validation (IV&V) activity.

Having proposed that, what changes are required to ensure this? Some of the key elements of an approach include:

- The identification of unchanged but operationally affected code,
- Development of automated code analysis tools,
- Software scenario analysis research,
- Exploitation of historical databases, and
- Shelf life used as a risk reduction factor.

Some of the interesting discussions that stemmed from this work focused on model checking and software scenario analysis. These are effective techniques to control state-space explosions and identify unsuspected behavioral properties. Is the tailored approach project specific? As with all IV&V practices, it must conform to the development environment for which it is intended. However there are core computer science principles that hold true for all projects, such as intractability and reachability, so it is conceivable that a robust verification and validation plan can lead to a reliable and efficient COTS intensive system.

Achieving interoperability and lower cost has resulted in sacrificing reliability, security, and maintainability. So we ask ourselves, when do we employ the use of COTS products? The majority says, "don't" when the system involves mission critical applications that cannot have insight into the COTS component, otherwise it may be feasible.

Session IV, V, and VI

The last three sessions contained similar themes and focused on vendor perspectives, user perspectives, and integration of COTS.

Have we evolved to an acceptable level that facilitates the efficient management of our entire IT environment? Practice does not seem to believe so. Management's role has positioned itself in such a way it manages platforms separately, relative to the domain of the module. The database component is managed separately from the network component, which could cause a redundancy in decisions, policy, and activities. Assuming an end-to-end management schema would enable the entire system to be optimally managed decreasing the amount of effort that can be exposed during development and deployment. This concept is actually an innovative approach that uses proven established techniques. Similar techniques are employed by system administrators to monitor the network in hopes of identifying and avoiding network outages. So promoting this concept to encompass the entire IT system is certainly a task worthy of further exploration.

As previously stated military insight into COTS products is extremely important. We have witnessed the evolution of civilian requirements imposed on COTS packages. We came from a paradigm that advocated quick development and a lot of functionality, in hopes of getting the product to market first. Currently it seems that civilian requirements are starting to resemble military crisis-mode requirements:

- Availability,
- Scalability,
- Reliability,
- Secure, and
- Interoperable.

A focal point is how to get industry to build highly reliable components.

COTS packages can be rendered and interpreted through an assessment of several qualitative dimensions. These dimensions include:

- Presentation interface: performance and universal access,
- Release compatibility: bug fixes, requirements, backward compatibility,
- Portability: interoperability, scalability, flexibility,
- Programming Interface: APIs and openness,
- Security Interfaces: integration with existing infrastructure, and

- Management interfaces: if you don't monitor then you can't manage.

This can be implemented multi-dimensionally, allowing scalability. This in turn can be used to find a "best fit" relative to system requirements in order to ensure its long-term applicability in a particular environment. However complexity arises when defining the "best fit" property and even determining the appropriate dimensions. This directly correlates to design spaces when we employ the use of design matrices to evaluate plausible architectures.

Conclusion

A third of the presentations addressed a subset of techniques that can be encapsulated as an appropriate system's assurance (Verification and Validation) paradigm. We have yet to identify which technique works best in certain situations. In addition we have seen the beginning of several analytical assurance techniques, which do not rely on the existence of source code.

COTS based acquisitions are not a cost-effective strategy for critical military systems. There are several reasons: limited assurance, latent bugs, no availability, not all the requirements are satisfied, and defined requirements need to be massaged for integration of a COTS package.

In light of military attitudes to long-term budgeting, no serious consideration will be given to COTS life cycle costs as opposed to initial procurement costs. The acquisition process is the biggest impediment for the use of COTS. Individuals need to understand the life cycle and the costs associated with it. The impediment is that managers must get a system out the door on time and within cost.

Open-source versus closed-source products, which are cost effective and risk adverse? We have little experience taking apart closed-source products, and we have better reliability via feedback mechanisms on open-source products. Critical systems need source code for formal verification techniques, and we lack appropriate tools to perform assurance techniques on closed-source products. In addition we need to move past our component assurance paradigm and maneuver more towards a systems assurance paradigm.

Recommendations

Employing COTS has been strongly correlated with saving money, a correlation that does not have strong empirical evidence. Additional assurance activities must be employed to achieve an acceptable level of risk of COTS products. The cost of these additional activities must not exceed the savings yielded from the use of COTS. Some of the associated drivers include the cost of acquiring the software, cost of upgrading the software, cost of software being unavailable for use, cost to repair the software, and cost of additional assurance activities.

In addition, the need for quality products that control military critical systems has recently surfaced. To answer this challenge industry needs to explore the development of a certification paradigm that quantitatively grades software quality by rigorous testing techniques with very well defined environmental assumptions.

The symposium revealed several evaluation and assurance techniques that can be employed to assist development of a COTS intensive system. However, there does not exist a formal system's assurance paradigm. A formal discipline could be conceived from the synergy of current techniques; however these ideas need to be explored and more importantly empirically validated. Some related topics that require further attention include:

- New testing strategies,
- Software Architecture Theory,
- Testbeds or agents that probe or monitor the environment where COTS products live,
- Model checking and reachability analysis must ensure that folding the state space retains the vectors of critical paths. Keep research dollars away from exhaustive testing approaches, path coverage analysis, and infallible proof finders for we know these are NP-complete problems,
- Risk management and/or risk analysis techniques need to evolve to incorporate COTS based systems, and
- Current COTS evaluation techniques need to be empirically proven.

There are a significant number of groups that have a vested interest in this topic. These issues warrant considerable attention, especially from a military perspective. This is why a "COTS User Group" needs to be established and administrated by NATO, in order to bring together members of academia, research, and operations. This would provide an optimal working environment to combat the problems, research issues, and failures of COTS based systems while disseminating the best practices, lessons learned, successes, and recommendations. A "COTS User Group" could easily model itself around the workings of the "Object Management Group (OMG)" or the "World Wide Web Consortium (w3c)". A few examples of issues the group could address are:

- Development of standards/recommendations, which focus on required COTS characteristics for mission critical systems,
- Evolving commercial standards (recommendations),
- Evaluation of development life cycle methodologies,

- Administering the development of a formal systems assurance paradigm.

As the industry embarks on newer technologies it is appropriate to assess the applicability of the different phases in the development life cycle and how the newer concepts and techniques affect them. As a result of several discussions it was concluded that defense procurement requires a drastic paradigm shift. Unfortunately the topic was not further entertained, ideally a model for the procurement process would be developed that can be easily configured in different environments. In addition there were a lot of concerns regarding the lack of efforts allocated towards configuration management and the operations and maintenance phases of COTS based systems.

NATO should conduct a follow up symposium within a few years, in order to identify COTS assurance technologies and concepts that have stabilized, evolved, or even disappeared. Keeping this symposium as a baseline would serve as a measure that reflects the effectiveness its' results have on employing COTS components in military applications.

Robust Nonproprietary Software

Peter G. Neumann
Principal Scientist, Computer Science Lab
SRI International, Menlo Park CA 94025-3493, USA
Neumann@csl.sri.com, http://www.csl.sri.com/neumann, 1-650-859-2375

The following text is a preprint of a position paper for a panel session at the IEEE Symposium on Security and Privacy, 2000 May 15-17, and will be included in the Proceedings of that conference. It serves here as a narrative explanation of the subsequent slides for my NATO talk, "The Potentials of Open-Box Source Code in Developing Robust Systems".

Our ultimate goal here is to be able to develop robust systems and applications that are capable of satisfying serious requirements, not merely for security but also for reliability, fault tolerance, human safety, and survivability in the face of a wide range of realistic adversities – including hardware malfunctions, software glitches, inadvertent human actions, massive coordinated attacks, and acts of God. Also relevant are additional operational requirements such as interoperability, evolvability and maintainability, as well as discipline in the software development process.

Despite all our past research, development of commercial systems is decidedly suboptimal with respect to meeting stringent requirements. This brief paper examines the applicability of some alternative paradigms.

To be precise about our terminology, we distinguish here between *black-box* (that is, closed-box) systems in which source code is not available, and *open-box* systems in which source code is available (although possibly only under certain specified conditions). Black-box software is often considered as advantageous by vendors and believers in security by obscurity. However, black-box software makes it much more difficult for anyone other than the original developers to discover vulnerabilities and provide fixes therefor. It also hinders open analysis of the development process itself (which is something many developers are happy to hide). Overall, it can be a serious obstacle to having any unbiased confidence in the ability of a system to fulfill its requirements (security, reliability, safety, etc., as applicable).

We also distinguish here between *proprietary* and *nonproprietary* software. Note that open-box software can come in various proprietary and nonproprietary flavors.

Examples of nonproprietary open-box software are increasingly found in the Free Software Movement (such as the Free Software Foundation's GNU system with Linux) and the Open Source Movement, although discussions of the distinctions between those two movements and their respective nonrestrictive licensing policies are beyond the scope of this brief analysis. In essence, both movements believe in and actively promote unconstrained rights to modification and redistribution of open-box software [2].

The benefits of nonproprietary open-box software include the ability of outside good guys to carry out peer reviews, add new functionality, identify flaws, and fix them rapidly – for example, through collaborative efforts involving people widely dispersed around the world. Of course, the risks include increased opportunities for evil-doers to discover flaws that can be exploited, or to insert trap doors and Trojan horses into the code.

A question for this panel is what are the roles of open-box software in developing robust systems, in light of (for example) the Internet, typically flawed operating systems, vulnerable

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

system embeddings of strong cryptography, and the presence of mobile code. An architectural subquestion involves where trustworthiness must be placed to minimize the amount of critical code and to achieve robustness in the presence of the specified adversities.

Will open-box software really improve system security? My answer is *not by itself, although the potential is considerable.* Many other factors must be considered. Indeed, many of the problems of black-box software can also be present in open-box software, and *vice versa* (for example, flawed designs, the risks of mobile code, a shortage of gifted system administrators, and so on). In the absence of significant discipline and inherently better system architectures, opportunities may be even more widespread for insertion of malicious code in the development process, and for uncontrolled subversions of the operational process.

We face the basic conflict between (a) security by obscurity to slow down the adversaries, and (b) openness to allow for more thorough analysis [3] and collaborative improvement of critical systems – as well as providing a forcing function to inspire improvements in the face of discovered attack scenarios. Ideally, if a system is meaningfully secure, open specifications and open-box source should not be a significant benefit to attackers, and the defenders might be able to maintain a competitive advantage! For example, this is the principle behind using strong openly published cryptographic algorithms – for which analysis of algorithms and their implementations is very valuable, and where only the private keys need to be hidden. Other examples of obscurity include tamperproofing and obfuscation. Unfortunately, many existing systems tend to be poorly designed and poorly implemented, with respect to incomplete and inadequately specified requirements. Developers are then at a decided disadvantage, even with black-box systems. Besides, research initiated in a 1956 paper by Ed Moore [1] reminds us that purely external *(Gedanken)* experiments on black-box systems can often determine internal state details.

Behavioral system requirements such as safety, reliability, and real-time performance cannot be realistically achieved unless the systems are adequately secure. It is very difficult to build robust applications based on proprietary black-box software that is not sufficiently trustworthy.

Further 1956 papers, by Moore, Claude Shannon, and John von Neumann, showed how to construct reliable components out of less reliable components. Later work on correct behavior despite some number of arbitrarily perverse Byzantine faults followed along those lines. In that context, building a fault-tolerant silk purse out of less robust sow's ears is indeed possible in some cases. But constructing more trustworthy secure systems out of less trustworthy subsystems does not seem realistic when the underlying components are compromisible, despite efforts such as wrapper technology and firewall isolation.

Whenever achieving security by obscurity is not the primary goal, there seem to be strong arguments for open-box software that encourages open review of requirements, designs, specifications, and code. Even when obscurity is deemed necessary, some wider-community open-box approach is desirable. For software and for system applications in which security can be assured by other means and is not compromisible within the application itself, the open-box approach has particularly great appeal. In any event, it is always unwise to rely *solely* on obscurity.

So, what else is needed to achieve trustworthy robust systems that are predictably dependable? The first-level answer is the same for open-box systems as well as closed-box systems: serious discipline throughout the development cycle and operational practice, use of good software engineering, rigorous repeated evaluations of systems in their entirety, and enlightened management, for starters.

A second-level answer involves inherently robust and secure evolvable interoperable architectures that avoid excessive dependence on untrustworthy components. One such architecture involves thin-client user platforms with minimal operating systems, where trustworthiness is bestowed where it is essential – typically, in servers, firewalls, code distribution paths, nonspoofable provenance for critical software, cryptographic coprocessors, tamperproof embeddings, preventing denial-of-service attacks, runtime detection of malicious code and deviant misuse, *etc.* [4].

A third-level answer is that there is still much research yet to be done (such as on realistic compositionality, inherently robust architectures, and open-box business models), as well as more efforts to bring that research into practice. Effective technology transfer seems much more likely to happen in open-box systems.

Nonproprietary open-box systems are not a panacea. However, they have potential benefits throughout the process of developing and operating critical systems. Impressive beginnings already exist. Nevertheless, much effort remains in providing the necessary development discipline, adequate controls over the integrity of the emerging software, system architectures that can satisfy critical requirements, and well documented demonstrations of the benefits of open-box systems in the real world. If nothing else, open-box successes may have an inspirational effect on commercial developers, who can rapidly adopt the best of the results. But I like the possibilities for coherent community cooperation, and have considerable hope for nonproprietary open-box software.

References

[1] E.F. Moore, Gedanken Experiments on Sequential Machines, *Automata Studies,* Annals of Mathematical Studies, 34, C.E. Shannon and J. McCarthy, eds., Princeton University Press, 1956. pp. 129-153.

[2] The Free Software Foundation Website is `http://www.gnu.org`, and contains software, projects, licensing procedures, *etc.:* The Open Source Movement Website is `http://www.opensource.org/`, which includes Eric Raymond's "The Cathedral and the Bazaar" and the Open Source Definition.

[3] Analytic tools for open-box source code include Crispin Cowan's StackGuard (`http://immunix.org`), David Wagner's buffer overflow analyzer (`http://www.cs.berkeley.edu/~daw/papers/`), @Stake's L0pht security review analyzer slint (`http://www.l0pht.com/slint.html`) and RST's ITS4 function-call analyzer for C and C++ code (`http://www.rstcorp.com/its4/`).

[4] The U.S. Army Research Laboratory (ARL) has supported my work on survivable systems, under contract DAKF11-97-C-0020. See `http://www.csl.sri.com/neumann` for a report for ARL together with a course taught in the fall of 1999 on developing robust systems. That Website also contains a paper on single-level multilevel-secure systems, N.E. Proctor and P.G. Neumann, Architectural Implications of Covert Channels, *Proceedings of the Fifteenth National Computer Security Conference,* Baltimore, Maryland, October 13–16, 1992, pp. 28–43, very much in the spirit of the thin-client architecture noted above.

The Potentials of Open-Box Source Code in Developing Robust Systems

Dr. Peter G. Neumann
Principal Scientist
Computer Science Lab
SRI International, Menlo Park
California USA 94025-3493
Telephone 1-650-859-2375
E-mail Neumann@CSL.sri.com

Commercial Off-The-Shelf
Products in Defence Applications:
The Ruthless Pursuit of COTS
NATO, Brussels, Belgium
4 April 2000

1

Abstract

We consider the development of robust systems that must satisfy extremely critical requirements such as security, reliability, safety, and overall system survivability. We examine the potentials of source-available software within emerging technologies such as the Internet, mobile code, and highly distributed architectures.

2

Acknowledgment of Support and Contractual Disclaimer

This material is based upon work supported by the U.S. Army Research Laboratory under Contract No. DAKF11-97-C-0020.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the U.S. Army Research Laboratory.

3

Fundamental System Needs

- Critical information systems and their applications must predictably satisfy stringent requirements such as security, reliability and fault tolerance, safety, robustness, and survivability in the face of many adversities, interoperability, evolvability, maintainability, ...
- Stringent requirements cannot be achieved without many factors, such as good development practice, operational security, etc.

4

Some Generic Problems in Software Development and Use

- Developing robust systems is inherently difficult.
- Today's "Best Practices" are inadequate and poorly applied.
- The *Common Criteria* approach is incomplete and unwieldy, but better than its predecessors. Protection profiles are likely to be incomplete.
- Operations management is difficult and riskful.
- These problems are ubiquitous.

5

System Realities Today

- Distributed and networked systems are riskful. Embedded systems are increasingly dependent on other systems rather than standalone.
- Personal-computer operating systems are typically flawed, bloated, inflexible, difficult to administer, not suitable for critical applications. Server OSs and firewalls are often misconfigured. Network protocols are inadequate. There are many risks.

6

NATO-Relevant Risk Examples

- Vincennes' AEGIS shootdown of Iranian Airbus: inflexible software, archaic hardware
- Patriot missile inaccuracies: system clock drift, resulting from requirement mismatch, bad clock software; fix arrived too late, by airplane, to avoid Dharan
- Sgt York gun: software flaws
- U.S. Navy: USS Hue City & USS Vicksburg software integration problems; USS Yorktown outage due to unchecked divide-by-zero

7

Proprietary-System Problems

- Many proprietary systems are not capable of satisfying critical requirements.
- *Black-box* systems (i.e., closed-box, in which source code is unavailable) hinder open analysis of system development processes and the resulting software quality, impede system integration, and prevent urgent on-site self-remediation.
- Lack of interoperability and composability often encourages inflexible monolithic solutions.

8

More Black-Box Problems: Need for Reverse-Engineering

- Interoperability, local code patching for flaw removal, maintenance, and other constructive purposes conflict with the original World Intellectual Property Organization (WIPO) Copyright Treaty prohibitions against reverse engineering, although some of those restrictions have been somewhat relaxed.
- The U.S. 1998 Digital Millennium Copyright Act is overly restrictive. So is the pending U.S. Uniform Computer Information Transactions Act (UCITA).

9

Black-Box Software: Benefits for System Developers

- Black-box code masks intellectual property, and impedes development of competitors' systems and dependent applications.
- Proprietary black-box business models are well understood.
- Consumer retention/loyalty is incentivized.
- For secure systems, attackers may be slowed down somewhat by "security through obscurity".

10

Black-Box Software: Potential Benefits for Users

- For proprietary systems, the identified proprietor is the supposed target for remediation, lawsuits, etc. (However, this is not necessarily a benefit, as remediation is often slow and lawsuits are even slower!)
- For black-box systems, there are very few significant benefits for users that cannot also be achieved with available source code, except possibly the delaying effects of security by obscurity.

11

Terminology

- *Open-box* code, i.e., *source-available*, is the antithesis of *black-box* code. Many examples of open-box software are found in the Open-Source Movement and the Free Software Movement (see next slides and cited Websites), with various distribution licenses. (Note: open-box software may or may not be proprietary.)
- Open Source and Free Software are not equivalent, although analysis of the differences is beyond the scope of this discussion.

12

The Free Software Movement's Free Software Foundation (FSF); `http://www.gnu.org`

In FSF, *free* implies *freedom to copy* and *freedom to change,* not necessarily *free of charge.* FSF incentivizes collaborative efforts and continual improvements. Founded 1984.

- The FSF General Public License (GPL) enforces *copyright* plus *copyleft,* where copyleft requires that redistribution (with or without change) must not restrict freedom to further copy and change.

Open Source Movement's Open Source Definition (OSD) `http://www.opensource.org/osd.html`

- Unrestricted redistribution
- Distributability of source code
- Permission for derived works
- Constraints on integrity
- Nondiscriminatory practices
- Transitive licensing of rights
- Context-free licensing
- No adverse affects on associated software

Open-Box Examples, e.g., Free and Open-Source Software

- GPL-ed: The GNU System with Linux (GNU Emacs, GCC, Gnome 2.0, Ghostview, GNUscape Navigator, gzip, Java packages, etc.), Free VSD; not quite GPL-ed software (Perl); non-GPL free software (Free BSD, X windows, Apache, LaTeX, Mozilla, Netscape JavaScript ...); Open BSD, Net BSD, Hyperlatex, Eazel's Linux graphical shell, ...
- Other licenses: MPL, QPL, ...

Open-Box Potentials 1

- Extensive peer review is easy and normal, and amenable to academics and other researchers.
- Peer analysis is capable of finding flaws and generating fixes rapidly.
- Open-box software is potentially more readily capable of incremental evolution.

Open-Box Potentials 2

- Users and administrators are potentially in greater control, because they may be able to obtain fixes and new features.
- People other than the original developers can add significant value, e.g., making systems more robust.
- Such software is often developed altruistically and less motivated by short-term cost-cutting.
- Open collaboration is easier.
- Software quality can be very high.

Analytic Benefits of Open-Box Code

The availability of source code for analysis (even if it is proprietary) enables application of analytic tools such as

- Crispin Cowan, StackGuard `http://immunix.org`
- David Wagner et al., Berkeley buffer overflow analyzer approach `http://www.cs.berkeley.edu/~daw/papers/`
- L0pht (now part of @Stake), slint `http://www.l0pht.com/slint.html`
- Reliable Software, ITS4 for C, C++ `http://www.rstcorp.com/its4/`

Open-Box Problems

- Many of the problems of black-box software are also applicable to open-box software, for example, the risks of mobile code.
- Opportunities may be more widespread for insertion of malicious code (e.g., Trojan horses) during development, and for operational subversions.
- Management may be needed across organizations, and is itself difficult and possibly riskful.

Open vs Closed Analysis: Seemingly Contradictory Views

- Easier access by adversaries to available source implies less operational security, because it is easier to find exploitable flaws in vulnerable systems
- Open box should be particularly important for the analysis and improvement of life-critical and ultra-reliable systems. Also, if a system is meaningfully secure, open specs and available source should not be of less benefit to attackers, which could give defenders a competitive advantage (for a change).

Available Source Is Only Part of What is Needed. There's More.

- Open-box source code shares many generic problems with black-box source.
- Much more is needed to make open-box systems robust, trustworthy, and predictably dependable.
- See my Website for background material on developing survivable systems and networks: `http://www.csl.sri.com/neumann/`

Generic Desiderata

- Discipline in development, software engineering, distribution, operation, evolution, evaluations, education, training, ...
- Inherently robust secure evolvable interoperable architectures
- Responsible operational support and configuration control
- Open standards for code, interfaces, composability, interoperability, distribution
- Contracts, liabilities, incentives: compliance bonuses, noncompliance penalties
- Sound business models for nonproprietary open-box software

Robust Architectures

- Architectures that avoid excessive dependence on untrustworthy components
- Thin-client user platforms with minimal operating systems, where trustworthiness is required only where essential
- Trustworthy servers, firewalls, distribution paths for software, provenance on all critical software; nontrivial user authentication, bilateral peer authentication, aggressive resistance to denials of service, better protocols, ...

More on Robust Architectures

- Nonsubvertible implementations of cryptography, used pervasively, including cryptographic integrity
- Run-time detection of malicious code and misuse
- Wireless applications and mobile code add some stringent further requirements.

Conclusions

- Nonproprietary open-box software (e.g., Free Software and Open-Source) is not a panacea, but has huge potential, with discipline and well-documented successes. (Discipline is similarly needed for black-box software, but is often lacking.)
- Open-box source could be particularly promising in efforts to develop dependable critical systems.
- Open-box successes can be an incentive to black-box developers, some of whom are already exploring such alternatives.

A Few On-Line References

- Peter G. Neumann: reports; testimonies; survivability course; RISKS materials; research papers, etc. `http://www.csl.sri.com/neumann`
- Free Software Foundation: software, philosophy, projects, licenses, etc. `http://www.gnu.org`
- Eric Raymond: Cathedral & Bazaar; Hallowe'en Documents `http://www.tuxedo.org/~esr/` and `http://www.opensource.org/`: "Open Source promotes software reliability and quality by supporting independent peer review and rapid evolution of source code."

BIOGRAPHICAL BACKGROUND

Peter G. Neumann is a Principal Scientist in the Computer Science Laboratory at SRI (where he has been since 1971), concerned with computer system survivability, security, reliability, human safety, and high assurance. He is the author of *Computer-Related Risks,* Moderator of the ACM Risks Forum (comp.risks), Chairman of the ACM Committee on Computers and Public Policy, and Associate Editor of the CACM for the Inside Risks column. He founded and 19 years edited the ACM SIGSOFT *Software Engineering Notes.* He is now a member of the U.S. General Accounting Office Executive Council on Information Management and Technology. See http://www.CSL.sri.com/neumann/ for Senate and House testimonies, reports, RISKS, papers, slides, etc.

Neumann taught at the Technische Hochschule Darmstadt in 1960, Stanford University in 1964, the University of California at Berkeley in 1970-71, and

most recently a course on survivable systems and networks at the University of Maryland in the fall of 1999 (see my Website for notes).

Neumann is a Fellow of the American Association for the Advancement of Science, the ACM, and the Institute of Electrical and Electronics Engineers (of which he is also a member of the Computer Society). He has received the ACM Outstanding Contribution Award for 1992, the first SRI Exceptional Performance Award for Leadership in Community Service in 1992, the Electronic Frontier Foundation Pioneer Award in 1996, the ACM SIGSOFT Distinguished Service Award in 1997, and the CPSR Norbert Wiener Award for in October 1997, for "deep commitment to the socially responsible use of computing technology."

Peter G. Neumann, Computer Science Laboratory, SRI International 333 Ravenswood Ave., Menlo Park CA 94025-3493 Telephone 650-859-2375, FAX 650-859-2844, neumann@csl.sri.com. http://www.csl.sri.com/neumann.html

WRAPPING THE COTS DILEMMA

Ian White
Defence Evaluation and Research Agency (DERA)
Portsdown Hill Road, Fareham, Hants, PO17 5AD
ENGLAND
iwhite@dera.gov.uk

Abstract

This paper first reviews the problems of using COTS, notably product assurance, vulnerability and product continuity. The concept of *wrapping* is then introduced. Conceptually, wrapping is a process to mitigate COTS limitations, and may be applied to any of the acquisition, design and implementation phases of a system. It is a fundamental assumption that COTS items being wrapped are not themselves amenable to any significant changes in their design or function.

1. INTRODUCTION

The title of the paper alludes to the common process of 'wrapping' inadequacies of COTS between added-value services that supplement the basic performance of COTS. Sometimes the wrapping is to add functionality; sometimes to limit poor functionality; sometimes the wrapping is cosmetic.

The use of commercial off the shelf (COTS)[1] components as the core elements of military information systems [command control and communications] is now widespread, because of their availability, low cost, and generally high levels of functionality. The military also increasingly expect a common look and feel between IT in the home, barracks and battlefield. However COTS elements are produced for the civil marketplace, and evolve in the context of a fine balance across commercial issues of:

- cost-competitiveness
- customer expectation of quality
- customer tolerance to shortfalls in quality
- lifetime in the marketplace
- commercial through-life support needs.
- time to market
- mechanisms for maximizing a future market share.

This balance of features is substantially different from those for traditional defence procurement, with huge advances in technology, and brutal conditions of the commercial marketplace, have resulting in the unprecedented growth in the capability and lowering cost of IT products. There are, from the military perspective, many inherent instabilities and inadequacies in this marketplace.

The military need to exploit COTS virtues and forgive COTS sins. The dilemma is that good comes with some degree of evil. The good things about COTS are taken for granted here. For insight into the future there is plenty of literature. For the author's views on future military communications see ref. [1].

2. COTS ASSURANCE - THE DILEMMA

If the military are to use COTS, and there is really no economically viable alternative in many situations, the military needs and commercial qualities must be reconciled. The heart of this issue is Assurance: to answer reliably the questions, what -

- does the product do?: is it properly specified?
- guarantees are given of its performance?
- is its reliability in performing this task?
  - as stand alone?
  - in an inter-operating military environment (both military and COTS hw and sw)?
- undeclared features does the product have (especially to 'illegal' inputs)?
- is the stability of the product in the marketplace?
  - its continuity of product?
  - the continuity of its function in replacement products?
- are the implications of new functionality on previous assurances

2.2 Manufacturers' Guarantees

Hardware guarantees are stronger than software guarantees, but neither is very strong from buyers' assurance viewpoint.

[1] To save space, in this paper COTS is variously used as a noun, (e.g. the use of COTS in military systems) or as an adjective (e.g. the use of COTS components).

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

Hardware

Guarantees typically provide for remedy of physical failures, including parts and labour, for the guarantee period. There is no specific *fitness for purpose* guarantee, although PC manufacturers make general compliance statements regarding preferred software systems. Whilst they give guarantees against physical failure, they give no guarantee against design defects, although they are often legally bound by national 'fitness for purpose' trading legislation, and health and safety considerations in this respect. For the military user with a long-term interest in function, design defects are the more important aspect.

Software

Main stream COTS software is not guaranteed by its manufacturers against functional failures; nor do manufacturers usually undertake to indemnify against software failures nor to correct all software failures. Examples of hardware and software guarantees from mainstream manufacturers indicate the problem faced by military systems users:

HARDWARE

Chrysler 3/36 Warranty*: "The basic warranty covers every Chrysler supplied part of your vehicle, except its tires . . . tires are covered by separate warranty"*

"The 'Basic Warranty' covers the cost of all parts and labour needed to repair any item of your vehicle . . . defective in material, workmanship of factory preparation. You pay nothing for these repairs"

Typical computer hardware warranties guarantee the hardware product against defects in materials and workmanship for a period of one year from the date of original purchase.

SOFTWARE

Software warranties are far poorer, typically disclaiming any liability for their use, nor giving any specific undertaking to correct faults that are discovered.

Why are guarantees so poor, especially for software? In the next section the expectations for software and hardware assurance of function are examined.

2.3 Software Assurance

The problem

The problem is that software now is like architecture in medieval times.

"If I were to build a bridge, I would, by using classical stress analysis, have figures for strength, elasticity, and a variety of other predictive capabilities to tell me how to build it. If my bridge design was wrong, it would probably, even during building, become evident that something was wrong.

If I wish to build a large software system - the bridge from the concept to the action - I have very little in the way of theory to do it. Building a software intensive real-time system is a task for which we have few exact tools and no fully effective means for predicting its performance. " [2].

Closed Source Software

Many common software products, for example office aids, operating systems, and compilers, are complex programmes typically of several million lines of executable code. Current software evaluation technology is not able to formally test such software for the conformance of its functionality against specifications, and often such detailed specifications are not available. Nor are specification methods sufficiently strong to ensure that specifications are themselves either complete or consistent. The primary COTS software testing process is:

<*beta* test using knowledgeable users, and revise>
then <sell and take customer feedback>
then <prioritize faults against cost/marketplace need>
then <revise, and issue marketplace patches[2]>.

It is commonplace for such test programmes to uncover literally thousands of faults. This is the number of known faults corrected, not the total list! The manufacturers' versions of the latter lists are not normally published, although WWW listings by user groups are common. Furthermore, virtually all commercial software products are delivered as compiled code tied to a proprietary operating system. The source code is not available; it is subject to neither third party peer amendment nor review.

It appears that the fault rate of large software programs:

- is never amenable to any analysis to determine its extent
- is a linear function of the number of skilled users who apply it.

The significance of faults is difficult to quantify, since a minor fault for one user, may have catastrophic consequences for another.[3] What is evident with large COTS software products is that system failures caused

[2] i.e. short blocks of replacement code for aspects of the SW that did not work correctly.

[3] The Excel bug in which one specific number was incorrectly represented internally is an example (i.e. 1.40737488355328, = 0.64; several related numbers also fail), Risks digest [3], Vol. 15, no 39.

by various known and unknown fault states are commonplace. A source code bug rate of 4 bugs /Kloc[4] is a commonly cited number [3], however this would imply that popular PC operating systems, which range from about 8 to 64million lines of executable code initially have circa 32,000 to 256,000 faults, which is not realistic. Most faults are excised in development and beta testing. The point is that even with fault rates orders lower than this, the residual fault rate is not trivial.

This commercial development model poses serious questions for users requiring functionality with high integrity. Whilst a judgment may be made on integrity after some initial bedding-out time, it is not in itself a guarantee of performance. The Windows 3.11® product is old but reasonably stable. The problem of this strategy is that such products are not supported for more than a few years, and W3.11® is now entering this unsupported threshold[5]. Unfortunately, manufacturers give no guarantee that the later products will be as reliable as the older ones. Generally they are not, because they are less well tested, and have additional functionality, less well tried, and sometimes unwanted.

The arena of reliable software is still a meagre green patch in the wide range of commercial software. Whilst there have been significant advances in the development of formal, and quasi-formal methods, their application still remains limited, mainly to relatively small programs, for overtly safety critical functions such as avionics fly-by wire functions.

Open Source Software (OSS)

The use of open source software, developed collectively by 'hackers'[6], allows a review process not just by users, but by programmers and users, who all have access to the source code. This level of scrutiny produces some excellent code, which is potentially of a substantially higher quality than closed source software. Military procurers have traditionally been rather wary of the culture of open software programmers and users. Ironically it could prove to be a more reliable source than proprietary products. Strong interest is now being shown in the open software UNIX ® operating system *LINUX* ®, which performs well, is well supported (including by some commercial organizations), and is gaining ground due to a strong degree of dissatisfaction with proprietary closed source products in some sectors of the operating systems marketplace. Perhaps even more remarkable is the success of *Apache*, which is the widely used Web server software. This is a function requiring very high reliability, which the OSS process has clearly achieved.

2.4 Hardware Design Assurance

Chip Design

The design of complex integrated circuit chips is also extensively supported by software processes. Because the implications of a failure in the chip design are severe, with one chip design being manufactured in millions, the manufacturers take considerable trouble to ensure that the functionality is sound. This includes a wide range of emulation testing. Notwithstanding this, errors do occur, a notable example being Intel with its early Pentium ® chip, which had a arithmetic processor error. The elementary functions of a computer chip are grounded in mathematical processes and operations, and the functionality of processors is a more constrained domain that the full spectrum of COTS software. Because of these factors chips, from both design integrity and physical reliability viewpoints, are normally orders better than software, although faults do still occur.[7] This allows us to conclude that:

**For most military systems,
the degree of assurance in chip designs from
major manufacturers is adequate.**

2.5 Vulnerability

The problem

The limitations of testing and completeness of function of COTS systems has been discussed. Associated with these weaknesses is a huge user and player community on the WWW, that disseminates information on system weaknesses, their cures, and of course the malicious exploitation of known weaknesses for disruptive purposes. Notable weaknesses of COTS, cited in [4], are:

- easier for non-specialists to make intrusions
- low-cost solutions ignore stronger protection issues - security, integrity and EMC.
- COTS are more vulnerable to viruses, Trojan Horses, Easter Eggs etc. due to the wide

[4] Kloc = 1000 lines of code

[5] It is many manufacturers' stated policy to only support the current and the previous release of software products.

[6] Not the colloquial meaning, but referring to software aficionados who devote their lives to developing open software.

[7] The command f0 0f c7 c8 causes the Pentium processor to halt; recovery requires power-down/power-up (Risks Digest, vol. 19/45).

dissemination of information on these over the Web.

- conformance is difficult to establish with systems needing frequent patches to remedy defects.
- low COTS price, and rapid 'time to install' creates procurement pressures to adopt COTS, even if the full extend of vulnerabilities is not clear.
- maintenance for COTS creates some potential problems in battlefield situations.
- unanticipated inter-system interactions
- accidental, or deliberate intrusions or erroneous functions.

A fascinating and extensive catalogue of these are presented in the *Risks Digest* web pages produced by the ACM [4].

Vulnerability Tests

Tests on COTS products can be 'black box' or 'white box'. The latter is where all details of the COTS item are available (software listings, specifications, performance results etc.). Clearly for COTS a black box approach is more realistic, although few black box tests are available for COTS testing.

Enhancing Robustness

Information warfare (IW) protections include increasing system diversity to provide resilience, and as a way of making any focused analysis of system weaknesses more difficult. This is of course counter to the main stream of COTS development where at any one time there is normally a dominance of a few or even one product for key functions.

3. WRAPPING THE DILEMMA

3.1 Principle

If the military are to use main stream products how can they do better than supinely accept the poor quality assurances, and vulnerabilities of commercial software? We cannot give any strong dictats to manufacturers, and so must somehow 'wrap the dilemma'. The wrapper lets the good things through, and traps the bad things. That is the principle; the practice is however very limited.

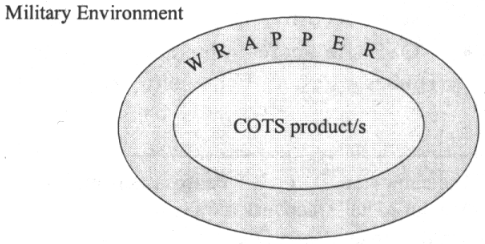


FIGURE 1 -- Wrapping COTS Functionality

The diagram illustrates the idea; the questions are:

- what needs wrapping and why?
- who produces the wrapper?

Wrappers fulfil two principal functions

- They trap poor or dangerous performance features (i.e. provide better assurance)
- They specifically enhance existing performance

A wrapper can be considered to be any, or all, of:

- An additional acquisition/procurement test
- Additional software
- Additional hardware.

3.1 Wrapping the Acquisition Processes

Product validation

It is possible in principle for the military buyer of COTS to test it himself to derive his own assurances of performance. This process is both difficult, and liable to be invalidated by the frequent changes made by manufacturers in their COTS products. Evaluation techniques include reverse engineering of closed source software, and using open source software, subject to certain tests. There are very few validation test suites for COTS systems. The primary source for such tests are national conformance testing centres, whose tests are primarily aimed at establishing standards compliance of systems interfaces.

Reverse Engineering

One possibility with some products is to reverse engineer from compiled code to evaluate the overall systematic functionality of the product. Although this is possible in principle, it has little general application because:

- for most programs it cannot be done economically[8]
- for large programs it cannot be done at all
- even if feasible, the scale of the analysis would require extensive resources to apply the tests
- any changes made as a result of such tests would
  - not be subject to any form of fault support from the manufacturer
  - probably be a breach of the manufacturer's copyright.

An alternative assurance strategy is to negotiate confidential access to the source code. For major

[8] One of the main targets for reverse engineering are computer viruses, which are normally quite small programs. Even these can prove difficult to analyse in this way.

COTS products most manufacturers see no commercial benefit in this, when set against the risks of their code confidentiality becoming compromised. Because major products are developed by large teams, at substantial expense, external assurance audits will require similarly large skilled teams. Accordingly a careful cost/benefits analysis for such assurance would have to be made. This leads to the second strong conclusion that:

In general reverse engineering is not practicable.

Exploiting Embedded Systems Software

Software for embedded systems is produced to a generally higher standard of reliability than for most software applications. Furthermore its design gives much more attention to economy of storage and processor demand than does general user software. One illustration of this is embedding *Windows*. Microsoft's *Windows CE* (*WinCE* ®) is a compact operating system, with some attractive 'military' features.

"While the industry trend (in OS's) is for successive OS releases to require even more hardware resources in terms of processing power, RAM, and disc space, WinCE has been written from the 'ground up' to operate with only the most basic hardware requirements"

Traditionally, Microsoft OS are supplied as a core kernel associated with a large number of supporting services, such as file systems and network support, all in a monolithic chunk. All these supporting services need to be stored on disc and loaded into RAM whether or not they are required by the applications running. The functionality available can be modified by adding or removing drivers for hardware options, but the kernel or removal of a service from say Windows NT®, would cause system malfunction with potentially disastrous consequences. Furthermore this would be considered a breach of the licensing agreement between the end user and Microsoft!" [5]

The virtues of a system built over a monolithic operating system are numerous and include:

- functionality limited to only that which is needed
- economy of required platform functionality (RAM, disc processor speed)
- far easier to configuration manage
- easier to undertake user driven test and evaluation of reliability/integrity
- deeper understanding of function available to user
- easier to assess vulnerability
- easier to apply formal or quasi-formal integrity/consistency tests to overall function.

The drawbacks are:

- less readily available range of functionality
- more expensive to develop
- may be difficult to expand to match the full range of the monolithic OS.

There are no standards here. The *EPOC*®[9] embedded operating system from Symbian (a joint venture company of Psion, Nokia, Motorola and Ericssen) is also a strong contender in this marketplace. The Symbian consortium is now attracting a very substantial international company following, and is seen as a major competitor to Microsoft's *WinCE* ® [6].

Safety Critical Requirements

If software must fulfill a safety critical function, then it must be structured for this task, and also run on high reliability hardware platform/s. We can take some ideas from fault-tolerant systems, for example running 'equivalent' software on different machines to provide parallel answers to important questions. In some cases running a hot standby can protect against hardware or software failures. These options require very high integrity choice mechanisms such as <majority voting on three processes> evaluations, or <failure detect - switch to standby>. The former requires that the same process is implemented (coded) in different ways to achieve some degree of independent failure mode. With COTS this is not generally possible, although some investigations have been made on the use of Unix systems from this perspective. These techniques are approximately three or two times respectively more expensive than stand-alone, and similarly more expensive to maintain. They may not be a cost effective solution for many defence support roles, but if COTS must be used for defensive critical roles, including much of C4I, then these options need to be considered much more seriously.

Unfortunately very little software written in the COTS marketplace is evaluated against these sorts of criteria. Nor are such results published as part of a assurance certification. This leads to the strong conclusion that:

Safety critical (SC) software cannot be provided from mainstream COTS. However more reliable software can be made using SC principles.

3.4 Architecture and Standards as Wrappers

[9] The name *EPOC* derives from the abbreviation of epoch, as this was regarded as being a new epoch in Portable Operating Systems.

"The difference between doctors and architects, is that doctors bury their mistakes"
Edward Lutyens (British architect).

Architecture mandates are the wrappers around the system design process. It is a cherished belief in military planning and procurement organisations that they can collect standards into a compendium, and even the relationships between standards, e.g. in profiles or structured APIs, and order the world to do their bidding. However to be effective an Architecture must meet the following conditions:

1. It must advocate a sufficiently precise set of standards, and related implementation processes
2. There must be a general community (industry, procurers, users) acceptance that these mandated elements are reasonable
3. It must be enforceable through indirect and evolutionary mechanisms as much as by policing.
4. It must be kept up to date
5. the procurement process must be matched to the timescales of the elements advocated within the architecture.

Historically many aspects of these mandates fail, often not through inadequacies in the standards themselves. The pace of COTS development, as part of this scenery, renders such mandates suspect. Using COTS has severe implications for those who seek to mandate procurement standards in IT. A fundamental reason is the standards generation process. To determine, write, and apply a new standard requires much discussion between peer experts, practical evaluations, refinements, and much deliberation over the writing of the standard to make it as clear and unambiguous as possible. This is inevitably a long process. There must be implementations those standards, which may be inimical to some companies' commercial objectives.

An Architecture mandate seeks to aggregate such standards, and in effect becomes a meta standard for a chosen set of systems (e.g. defence IT systems). It is axiomatic that to undertake this task requires a range of expertise that is both technically wide and deep. Defence architecture teams are never greater than a few people strong, they all have some technical weaknesses, and as the objective is an aggregation of standards, a full understanding of their interactive use is needed. In IT, this process is further aggravated by the rapid rate of change of products, usage styles and the continuous emergence of de-facto standards. This will always raise questions about the rational of any defence architecture mandate. Hence,

Architecture mandates do not address the problems of using COTS in military systems.

3.5 Wrapping COTS Products

Detecting faults and undefined states

Apart form seeking assurances that the product does what it should, there are also many weaknesses in products when they are used in ways that are not within the specification. Such 'out of range' states may be entered by accident, or by malicious design. COTS products are different from bespoke designs. Some manufacturers produce quite detailed specifications, others do not; some even charge for additional information describing the product. Most consumer software for example comes with no written documentation, other than installation instructions, licensing dictats, and installed 'help' facilities, that are often poor. The handbooks describing the product are an additional secondary market. No guarantee given that such descriptions are complete or fully accurate.

In addition, products often come with undeclared features (Easter eggs), that are used by the developers, and which the vendor may not wish the customer to access. These may be left for maintenance purposes, or as a deliberate act of configuration control ("this version passed acceptance tests just in time so don't change anything else!"). Such additions may even be gratuitous, as with the flying game in Microsoft *Excel* ® [10].

If we are to use such products in defence applications we need to know a lot about their behaviour. Apart from faults where the application does not do what it should, for a legal input parameter set, there are always many inputs sets which are illegal, and for which the system state is undefined. Note that manufacturers normally do not consider such results to be faults. Such states often cause the system to crash, so for any high reliability system these states need to be trapped. The scope for users, and hosted applications to invoke such values is evidently high.

We have already discussed design type checks on the procurement of systems. A better approach is to embed such COTS elements within some form of wrapper. The technology of wrapping is still developing, and is one where the military R&T need a greater and more strategically determined focus. The standard way of testing a process/equipment is by black box testing, where it is accepted that its detailed inner workings are not known. Performance is based on the interpretation of the relationships between inputs an outputs.

Black Box Testing - CMU's Ballista

A common problem with many operating systems is that parameter entries are not properly delimited. The extent of this problem is nicely illustrated by the work done by Carnegie Mellon University (CMU) on the *Ballista* Project [7], [8]. This project exploits the fact that many system crashes/lock-ups are caused by illegal values of module parameters. Accordingly they determine a parameter range for each operating system (OS) module (both legal and illegal values) and then test these, both individually and in nested iterations, with combinations of these parameter values.

It is part of the *Ballista* philosophy that the code for the module under test (MuT) is not available. Each module is characterised only in terms of parameter and data types required. Failure is defined informally according to user accepted crash/lock-up conditions. No special test harnesses/scaffolding is needed. *"The set of test cases used to test a MuT is completely determined by the data types of the parameter list of the MuT and does not depend upon a behavioral specification. "[8].*

Ballista has been used to evaluate *POSIX* OSs by testing 233 different *POSIX* calls. Input parameter values applied over all value combinations over various *POSIX* OS systems showed an average failure rate of about 15%. (i.e. failure count/{no. of input parameter combinations attempted}). The lowest rate was 10% for AIX 4.1, the highest 22.7% for QNX4.24.

Knowledge of these failure states can be used to define a wrapper that traps known illegal values, or even legal values, which cause failure. If the test results used for figure one are re-evaluated with all non-exception (legal value) tests removed, the resulting normalised failure rate is nearer 30%.

Using the diversity of these different OSs to provide a more robust response has also been investigated but shows little advantage unless a high level (many OSs) are used.

A wrapper family based on these results is currently under development by CMU. Although there may be speed penalties imposed by the use of wrappers, the reliability of the operating system can be dramatically increased.

[10] To access this game on *Excel-97* : open a new worksheet and press F5; in dialogue box type X97:L97; click OK; press tab; Press Control-Sh AND click *Chart Wizard*. Excel's *Mountain World* is revealed, and upon one of its peaks is a list of credits for usability testing! The game in Excel 95 is more elaborate. In *Powerpoint-97®* select *Help/about Microsoft Excel*; click the *Powerpoint®* icon in the dialog box; a list of the programmers appears.

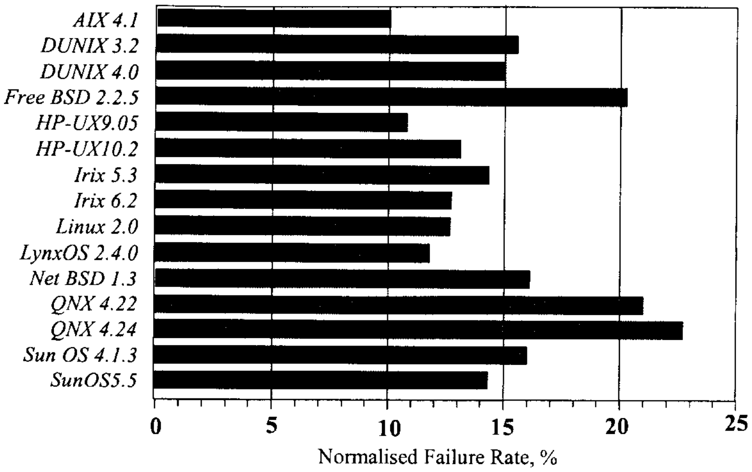


Figure 2 -- Ballista Testing: Posix System OS 'failure' rates[11]

It is salutary to note that as many manufacturers argue that such exception inputs are not faults, but user mistakes, they are not at fault. The corollary to this is that manufacturers may not include such states in their bugs databases, and may make little or no attempt to correct them.

3.6 Adding Functionality

Principle

Often in military systems the basic functionality provided by COTS systems is inadequate, or even non-existent. It is often possible to wrap a to COTS product, or service, with additional equipment to provide the additions required.

Crypto

The commonest military example of adding functionality is that of cryptographic devices. For example several ISDN cryptos are available that wrap the ISDN 2B+D channel, applying serial cryptography to the 2B stream, and applying signaling filtering to the D channel to prohibit data transmission and to constrain the range of signaling allowable.

The NATO KG81 applies a similar encryption to E1 (2Mbits/sec) trunks. Cryptographic devices of this type are both logical and physical wrappers.

Software insertion

Software wrapping, for example between a communications protocol and the API poses more significant problems, because the logical

[11] (from ref [8], courtesy Professor Philip Koopman, CMU Ballista Programme).

boundaries of software are less well defined, and more readily covertly subverted, there is far less confidence in the minds of security accrediting authorities about the use of software insertion mechanisms. This can be mitigated by related physical separations, for example dual processors, with accredited inter-links. Unfortunately this suspicion is not reflected in civil manufacturers' designs, so that security wrappers are often GOTS items, produced and distributed under specialist controlled conditions.

Physical Robustness

Another common need for additional functionality is to provide physical robustness, by additional packaging of equipment and by screening to mitigate the effects of radiation and Tempest. This is perhaps the original wrapper.

Enhancing performance

In some military applications the performance of COTS systems may be broadly what is needed, but with poor performance. Current IP routers, and ATM switches have poor performance in the present of data transmission channels that have poor error rate performance. This latter feature is a common characteristic of tactical military communications. To enable COTS products to be used frame and cell hardening is needed, whereby the link error rate is improved by error correction coding. This allows the commercial switches to be used, without any modification. There are now several products on the open marketplace that perform ATM Cell Hardening, including Comsat, SRC, Thompson and Marconi.

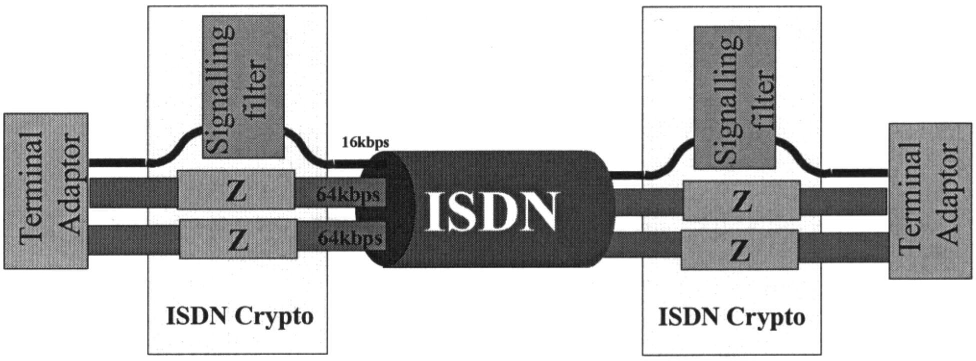


Figure 3 -- ISDN Crypto 'Wrapper' for 2*64 + 16 data channels

3.7 Interoperability Standards

Wrapping Military Legacy

Interoperability Standards exist in both the civil sector and NATO. An inverted strategy can by taken within military systems by wrapping them in a civil standard interface. This is being done with some systems adopting the various ISDN interfaces for military systems interfaces. For network management interfaces the process of wrapping legacy capabilities with common object interfaces is also being pursued.

There are a number of exciting developments in the civil sector on interoperability, all of which offer wrapping opportunities for military IT systems. Common examples are the use of ISDN BRA and PRA interfaces in military systems, the use of HTML web page based interfaces between systems (e.g. for network management), and more promisingly the use of CORBA techniques for integrating a wide range of legacy and proprietary systems.

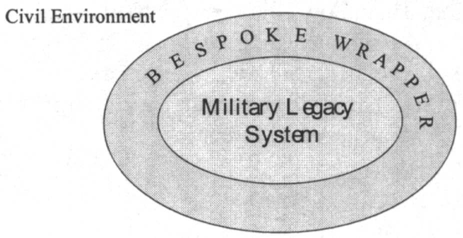


Figure 4: Wrapping Military Legacy, to Civilize it

Interoperability Opportunities

An interesting idea within the interoperability domain has been to develop a protocol language which allows 'on the fly' translation between protocols. This idea now has a real instantiation with the introduction by Sun Microsystems of the JINI ® concept. Here different computer devices are able to tell other elements in a computer network just what they are, and what they can do, using Java as the basis for this dialogue. A further development is to apply this idea not just to computer system interactions, but to a wider range of electrical elements. Wireless interconnection of these elements is within a programme called *Bluetooth* [9]. It provides wireless communications between element at ranges up to 10m, using an unlicensed region of the radio spectrum. In the office world this would link not just computers, but a very wide range of equipment, fax machines, telephones, shredders etc. In the home this linkage would be to all electrical equipment, the television, cooker, fridge, toaster, etc to advise the owner of the state of his home. It is the management infrastructure for the intelligent environment.

System management capabilities, deriving from the wide ranging work on enterprise resource planning/management, and the Telecommunication Management Forum's work on process definition, are further examples of opportunities for establishing usable military interfaces.

4. STRATEGIES FOR SELECTING COTS

4.1 Military-Civil Convergence

Mobile data services

What starts life as a 'Military Feature' in time often becomes a civil one. The need is for *'the office in a pocket'*, leading to increased data capacities for mobiles as well as more standardised forms of compact file structures for office utilities, for example 'lean' web page formats.

Security

The internet serves several hundred million subscribers, mostly without significant security features. This is a great untapped marketplace, since much of the internet will ultimately require a range of security services for its transactions, notably e-commerce both business to customer (B2C) and business to business (B2B). Already the ubiquitous virus has led to vulnerability protections becoming widespread.

Security mechanisms are now widely understood in the open literature and the knowledge is worldwide. Even unbreakable codes are readily within the reach of even modestly sophisticated users. The result of this strong drive towards powerful security will lead to a convergence of the civil and military security communities.

4.2 Which Products?

Four Key Questions

Clearly COTS products should be chosen because they perform a function that the military needs. The questions to be asked are about assurance and vulnerability, thus:

- Can the product be tested for (what it should do, and for what it should not)?
- Can it be replaced?
- Will it inter-operate with other systems/elements?
- Can it be wrapped without any internal modification?

It is a new challenge for military R&D to determine what research is needed to understand and best exploit the COTS marketplace. Since the marketplace overall is immense, an in-depth coverage is simply not feasible. So what should be studied? The focus must derive from both military functionality and procurement concerns such as those listed above.

Systems Integration

Many of the preceding issues have been concerned with understanding the function and performance of specific COTS products. To create systems comprising these elements requires:

- a design process
- systems integration and testing
- through-life support.

Each of these processes is different from those of old, where the design, even of much of the hardware, was bespoke. Specific performance limits, the implications of the changing COTS products in the time between design and integration, require that the process is much more fluid. Paul suggests that the dynamic pace of both requirements and technology invalidates many of the current design paradigms. There is a need for a developed concept of 'living systems', and design and support mechanisms to relate to this idea, rather than the idea of the system as a gestation aiming at a future 'fixed point'. He argues that the concept of a target 'fixed point' that the designers are trying to achieve is an obsolete, and dangerous concept within IT systems [10]. Certainly there is much to be said for integration being considered as in integral part of the design process, and the system itself being structured to permit its' evolutionary development.

A useful 'lessons learned' summary of COTS integration experience is given by Fox and Lantner [11]. A summary of some of their points is:

- COTS design results naturally in accelerated development, which precipitates an early start of integration testing.
- To be able to undertake integration testing test harnesses of various sorts are needed to simulate various aspects of the operational environment. The development of these can be a serious cost and time constraint.
- Maintenance on identified problems is by the COTS vendor, and may not meet the integrators needs. Problem investigation and identification and major parts of the integration task.
- the systems integrator must understand how products are configured (configuration files for specific products need to be rigorously controlled).
- Through-life support for COTS requires planning for replacement elements, at frequent points in the life-cycle, and this requires very strong configuration control.

5. ACKNOWLEDGEMENTS

This paper builds on earlier NATO IST contributions on the use of COTS by John Laws, Randall Schumaker, Major W. Tack, and various discussion and papers from within the UK's Defence Evaluation and Research Agency (DERA).

6. REFERENCES

[1] White, I, *Future Military Communications - A Preview*, unpublished paper.

[2] White, I., *Science in C3, Friend of Foe*, Joint Director of Laboratories Conference on C4I, Monteray, Cal. USA, 1994.

[3] Smith, C., *Win2000, 63000 Bugs*, Risks Digest, ACM publication; 19 July 1994, Vol. 20, no 80 (see also 81 and 82): all volumes available on the Web at: http://catless.ncl.ac.uk/Risks

[4] Tack, W., Maj., *C4I Vulnerability Using COTS*, IST Panel; 'Issues' Paper, Oct 1998.

[5] Watts, C., *Embedding Windows*, IEE Review, November 1998.

[6] *Overview of EPOC*, http://www.symbian.com/tech/papers/overview/overview.html

[7] *The Ballista Project*, Carnegie-Mellon university, http://www.cs.cmu.edu/afs/cs/project/edrc-ballista/www/

[8] Koopman, P., and De Vale, J. *Comparing the Robustness of POSIX Operating Systems*, 29th Annual International symposium on Fault Tolerant Computing, 13 - 18 June 1999, Madison Wisconsin, USA.

[9] *Overview of Bluetooth*, http://www.mobilebluetooth.com/

[10] Paul, R., *Why Users Cannot Get What They Want*, ACM SIGIOS Bulletin, Dec 1993, Vol. 14, No 2, pp 8 - 12. (also on University of Brunel (UK) web site).

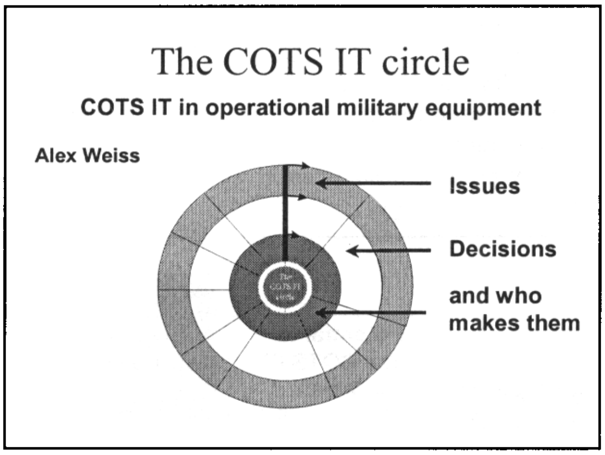
[11] Fox, G. and Lantner, K. W., *A Software Development Process for COTS Based Information System Infrastructure*, http://www.stsc.hill.af.mil/crosstalk/1998/apr/process.html

The COTS IT Circle

Alex Weiss,
Defence Engineering Group, Department of Mechanical Engineering,
University College London,
4th floor, 66-72 Gower St, London, WC1E 6BT, United Kingdom.
e-mail a.weiss@ucl.ac.uk

An examination of the issues raised by using COTS IT in operational military equipment, the decisions that need to be made and who has to make them.

The COTS IT Circle shows the issues raised when using COTS IT in operational military equipment. It looks at the decisions to be made and shows who has to make them. It starts by examining the main operational issues.

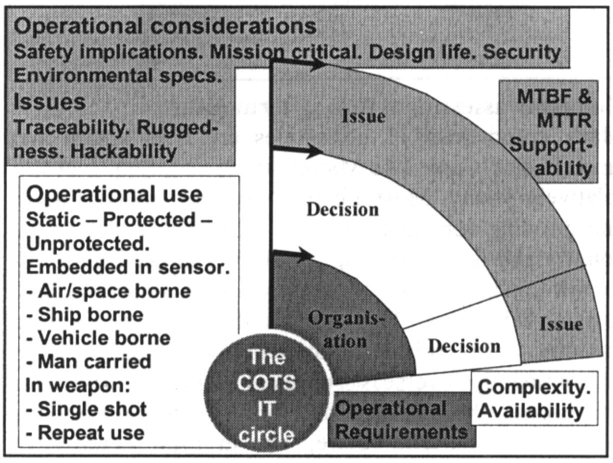

The COTS IT circle

COTS IT in operational military equipment

Alex Weiss

Operational considerations

There are a number of operational factors that need to be considered at the start of any project. Four of them are particularly relevant to the utilisation of COTS IT.


Operational considerations
Safety implications. Mission critical. Design life. Security
Environmental specs.

Issues
Traceability. Ruggedness. Hackability

Operational use
Static – Protected – Unprotected.
Embedded in sensor.
- Air/space borne
- Ship borne
- Vehicle borne
- Man carried

In weapon:
- Single shot
- Repeat use

MTBF & MTTR Supportability

Complexity. Availability

Operational Requirements

Safety implications

Does the use of the equipment have any safety implications and is it classified as a safety critical system? Such criteria demand the provision of a safety case; something that is difficult with a COTS IT-based system. Traceability is one of the key factors missing from virtually all COTS IT and there is little indication from the main suppliers that this situation is likely to change.

Mission critical

While the equipment may not have a safety role, it may well be critical to the completion of its mission. Such roles include COTS IT embedded in key sensors and weapons. Clearly, duplication of low-reliability parts rapidly increases the chances of successful functioning throughout any mission.

Total design life

The length of the design life of almost any piece of military equipment is far longer than for most of its civilian counterparts. A few aircraft and ships may remain in service, after a mid-life update, for as long as fifty years and a period of twenty-five years is commonplace. COTS IT, on the other hand, is obsolescent in eighteen months, obsolete in three years and mostly replaced within four years, even by the most cost-conscious users.

Interoperability

Interoperability mainly revolves around the question of standards and in the case of COTS IT, these are largely de-facto standards. However, experience with popular programs such as Microsoft Office shows the difficulty of interoperability between different versions of the same programme. The problem may become significantly harder and more expensive to deal with when upgraded COTS software has to interface to custom military hardware with an interface to the original version of the COTS software.

Operational use

The type of operational use will affect the type of requirements facing any COTS IT used. Almost all military equipment may experience a wide range of different environments depending on the particular application.

While, historically, defence specifications have carefully defined these different environments, COTS IT has had less care taken in specifying the environment in which it is to be used and suppliers to the military have made extensive use of wrapping to protect what would otherwise be very vulnerable items. The main area of COTS hardware differentiation has been between portable battery-powered items and mains-powered static ones.

Static

By far the most benign environment is that which is static and protected. Examples include COTS IT installed in permanent defence ministry buildings and in headquarters bunkers. More demanding is the use equipment in a static but unprotected environment. These are increasing found when the armed forces are deployed overseas and will locate equipment in existing

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

buildings that may or may not have central heating, air conditioning or sealing from damp, dirt and dust.

Embedded in sensor
Sensors themselves may require some or a great deal of IT to function successfully. At the top end are the requirements of electronic warfare sensors, while at the bottom end are relatively simple sensors such are thermal imagers. The embedding may well take the form of wrapping, but consideration also needs to be given to the likely deployment of the sensor. Big air defence radars are unlikely to be moved, while a sonar buoy may have to withstand impact with the sea when dropped by an aircraft, not to mention exposure to the maritime environment.

Air/space borne
The nature of the unprotected environment in aircraft is severe. Low temperatures and pressures are often allied to high vibration and 'g' levels. There is, however, increasing pressure on the aerospace industry to provide pressurised, temperature-controlled compartments for avionics equipment, which is then mounted on suitable anti-vibration mounts.

The situation in spacecraft can be even more severe. Not only must the equipment survive the launch, but it must also cope with the wide range of temperatures, vacuum conditions, micrometeorite impacts and various types of radiation found in space.

In both cases, weight and volume are major considerations as are heat generation and power consumption; the last particularly in space applications. Furthermore, in aircraft, the production of poisonous fumes in the case of fire in the air must be avoided for the safety of the crew.

Ship borne
Areas of problem for COTS IT arise on board ships and submarines for a variety of reasons. The first is the presence of a salt-laden environment. Low frequency of vibration is a particular issue and equipment must be able to survive exceptional levels of shock should the vessel be hit by enemy action. The generation of smoke or poisonous fumes must be avoided, particularly in the case of submarines. While power consumption is less of an issue than in aircraft, the generation of excess heat below decks often calls for a water-cooled heat exchanger. A particular issue is the need for mission availability, which can be for 90 days or more, relying only on on-board support for maintenance.

Vehicle borne
Any equipment installed in a vehicle leads a tough life. Exposed to the ravages of the weather, it is also expected to survive very high levels of shock and vibration. Maintenance also usually takes place in a less than ideal environment.

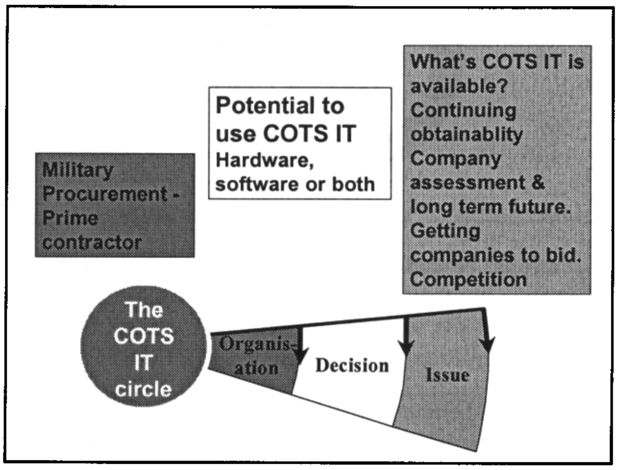
Man carried
Any man-portable equipment has to survive a large degree of rough and tumble, particularly in wartime. The elements, dust and mud, being dropped or thrown into a vehicle are all the lot of man-portable equipment. Size, weight, silent operation and low power consumption are important issues.

In weapons
Weapons include guns, rockets, guided missiles, mines and torpedoes. Almost all experience serious stresses at launch.

Single shot
Any single-shot equipment must operate the first and only time that it is used after a storage period that may last for decades. The increasing trend towards 'sealed' rounds avoids any checking or maintenance.

Repeat use
The repeated shock on any item of equipment that is part of a multiple-firing weapon system is bound to be severe.

What COTS IT is available
People working in defence ministries and for defence contractors are finding it increasingly difficult to keep up to date with what is being offered in the market place. The main reason for this is that the range of products is increasing rapidly as the market grows and this is allied to speedy product obsolescence; the result of the rapid changes in technology.


What's COTS IT is available?
Continuing obtainablity
Company assessment & long term future.
Getting companies to bid.
Competition
Potential to use COTS IT
Hardware, software or both
Military Procurement - Prime contractor
The COTS IT circle
Organis-ation
Decision
Issue

Company assessment & long term future
Many commercial IT companies are both young and small in size. Some like Microsoft are enormous, yet still relatively young. Most are following the industry norm and growing very fast. Few are located just in a single country and it is commonplace for the larger companies to sub-contract work to employees in countries like Russia and China. This implies that either a defence ministry or its prime contractors must manage these predominantly overseas suppliers, with the risk that support may be embargoed in times of tension or war. As for the long-term future, who in 1980 would have predicted the fall from the top spot of IBM?

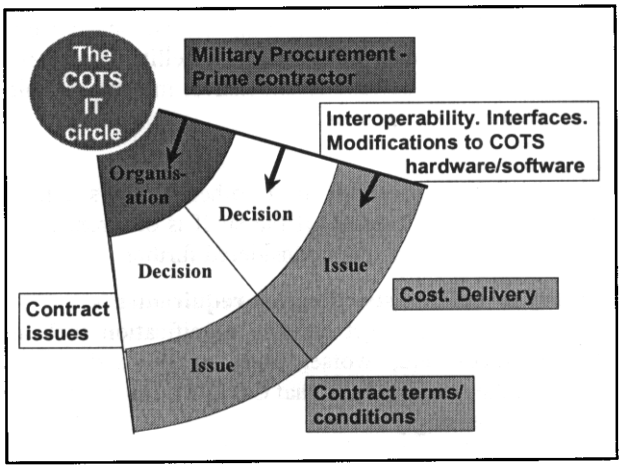
Getting companies to bid
The commercial IT market is huge while the military IT market is very small; representing only one percent of the total. This, in itself, is not a great incentive for commercial companies to bid for military work. The attractions of bidding are further reduced by the

aggravation involved in the bidding and contracting hurdles put in place by government military purchasers.

Competition
There are monopolies, or near monopolies in some areas, and the dominance of Microsoft and Intel in the software and microprocessor markets is well established. This can mean that it is sometimes hard to find true competition and this gets worse, once a project is locked into a particular IT solution. In the case of hardware, there is a plethora of 'IBM PCs' but the performances of these look-alikes is by no means the same. Nor are they necessarily always suitable in terms of form, fit and function.

Continuing obtain ability
In the time from deciding the content of a tender to the award of a contract, an item of COTS IT hardware or software may no longer be available. A year is a long time in the commercial IT industry, but only a short time in the military acquisition process, the more so if platform (ship, tank or aircraft) time scales are taken into account. It may well be that the COTS item can only be obtained as an upgraded version, which may or may not meet the requirement. It is difficult to keep up to date in terms of knowing what COTS IT is on the market and matching this against what will be needed. For some military requirements, 'Milspec' equivalents will be essential and COTS IT may not be able to be wrapped or otherwise modified to meet these requirements. In these cases, it will be essential to fund specifically these military areas of IT.

Potential to use COTS IT
It is at the earliest stage that a decision must be made on the possibility of using COTS IT. Such a decision is likely to impact back into the equipment specification, which must reflect its proposed use.


The COTS IT circle
Military Procurement - Prime contractor
Interoperability. Interfaces. Modifications to COTS hardware/software
Organis-ation
Decision
Decision
Issue
Issue
Cost. Delivery
Contract issues
Contract terms/ conditions

Delivery
The delivery time of COTS IT is remarkably short. It may often be literally off-the-shelf and may, in any case, be too quick for the purchaser. On the other hand, it may no longer be available when required. Continuity of supply and build standard are both issues that cannot easily be resolved. Furthermore, once the COTS IT has been delivered, there may be significant system integration problems, both in terms of the need to protect hardware, and in both hardware and software interfacing.

Cost
There is no doubt that bespoke systems are now largely unaffordable from the current levels of defence equipment budget of the industrialised nations. There is a significant cost of testing COTS IT to prove that it is 'problem free', and this may need to be added to the actual purchase price. It should be noted that the US DoD is carry out a great deal of COTS IT testing at its own expense.

The life cycle cost implications of using COTS IT are largely unknown because no major platform or system has had time to pass through more than a fraction of its life since COTS IT started to be used.

Competitive policy tends to be anti-COTS IT, since once a particular supplier of, for example, some software has been chosen, that supplier will be the sole potential supplier of software upgrades.

Modifications
COTS IT is available at remarkably low prices for standard items, though these low costs rise to ridiculous levels if modifications are demanded. It is clear that the initial operational requirement must reflect the potential for COTS IT use if major modifications to standard items are to be avoided.

Interoperability and interfaces
The need for interoperability between different COTS IT-based equipment and between COTS IT-based and bespoke military equipment is largely an issue of cost. Careful though about the use of interfaces early in any programme is key to minimising costs at later stages.

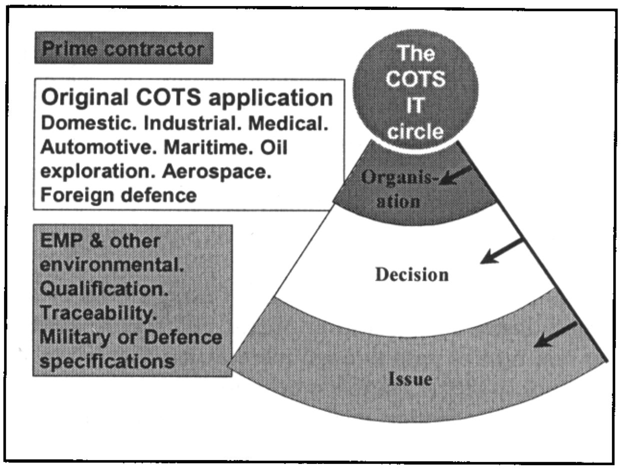
Contract terms and conditions

The terms and conditions of contract offered by defence ministries are not attractive and are often unacceptable to COTS IT suppliers; and this is particularly true of IPR. With most COTS IT suppliers located in the US or Pacific Rim, these firms are usually reluctant to send a negotiating team to another country for what they see as a contract in an irrelevant or sidelined market. Defence prime contractors are in no position to flow down their customer's terms and conditions to commercial IT companies and face a 'take it on our terms or leave it' attitude. Alternatives are to buy or licence software.

Original COTS application
It is clear that COTS IT is produced to many different standards, depending largely on the original application. A number of different commercial sectors are considered to examine how they vary.

Domestic
Equipment designed for use in the home, such as the microprocessors and other integrated circuits found in washing machines, microwave ovens and video recorders operates in a relatively benign environment. It is usually static, with a narrow operating temperature range. Electro-magnetic compatibility is important but design life for all white goods is only five years. Mobile

products, such as digital and video cameras are pushing a trend towards increasing physical robustness.


Prime contractor
Original COTS application
Domestic. Industrial. Medical. Automotive. Maritime. Oil exploration. Aerospace. Foreign defence
The COTS IT circle
Organis-ation
Decision
Issue
EMP & other environmental. Qualification. Traceability. Military or Defence specifications

Commercial
Information technology designed for commercial use operates in a similar environment to domestic equipment but is usually required to be more reliable as the consequences of failure, for any reason, usually have financial implications. In addition, the consequences of a hacker accessing, for example, a banking or other financial system can be extremely serious. Anything from a PC-based system, through a server to a mainframe system may be crucial to the operation of any commercial concern.

Industrial
Industrial systems, particularly those operating on a continuous basis, such as production-line equipment in a steel or glass works, or those undertaking robotic tasks, must have the highest availability. Again, the environment can be remarkably demanding and wrapping of delicate electronic equipment is widespread.

Medical
Information technology may just be commercial or industrial adapted for a medical role but it may operate equipment, such as X ray machines, where incorrect operation has the capacity to kill. Thus some applications involve safety critical operations; a fact which may be particularly applicable to some military requirements.

Oil Exploration
Many of the areas where oil companies are exploring for new finds have hostile environmental conditions. These include rigs in the North Sea, South Atlantic and Gulf of Mexico, land-based equipment in Alaska and Siberia, as well as tropical and desert regions. The application of IT in this industry has provided some exceptional wrapping issues, with a salt-laden atmosphere common and extremes of temperature as wide as any experienced by military equipment.

Nuclear
Much of the IT equipment operating in the nuclear industry is actually used to control or monitor the operation of nuclear reactors. This is a very safety critical function and equipment failures or crashes cannot be tolerated. Historically, custom-built systems were the norm but, as with military IT, the nuclear industry is being forced to embrace COTS IT.

Automotive
Not only are some automotive applications of IT safety critical, such as drive by wire, but the equipment also has to operate in tough environment. A wide range of operating and survival temperatures and humidities is essential and the mobile environment implies a high level of shock and vibration. Engine management systems are often fitted close to high-revving internal combustion engines and must operate reliably through the design life of the vehicle – typically ten years. Much of the standard IT used on commercial vehicles is already being applied to military versions as well as to new military vehicles.

Maritime
In some ways more benign that the automotive environment, the salt atmosphere and low frequency vibration levels must be survived. Some systems again are safety critical, particularly those that control the engines and steering, while others, such as navigation systems may be mission critical. With long periods spent at sea, the only maintenance possible is that which the ship's engineering staff can carry out using on board spares.

Aerospace
The ultimate safety critical environment, civil flight control systems have to survive a pretty tough environment, in many ways similar to that found in military aircraft. These and other IT based systems can be exposed to trying conditions including a very wide range of temperatures, low pressures and a broad span of vibration and humidity.

Spacecraft, while less safety critical, have exposure to a wide range of severe environmental conditions both during launch and in the hostile emptiness of space itself. Furthermore, the cost of getting a satellite into Earth orbit or beyond is extremely expensive, making reliable performance a key criterion.

Foreign defence
Military equipment supplied to other nation's armed forces and then purchased off-the-shelf is different from normal COTS IT and is not considered further.

Military or defence specification requirements

COTS IT does not meet defence specification such as US mil specs and, worse, there is no audit trail. However, the specifications that COTS IT can meet are:

Increasingly severe.
Usually not guaranteed by the supplier.
Often better than the supplier suggests.

EMP and other environmental
Hardware is not radiation hardened and for many military applications, ruggedness still an issue, leading to the need for wrapping to provide the required level of protection. While environmental requirements in the commercial sector are increasing, and much COTS IT is built to avoid RFI, there is little actual testing and there

are no Tempest-proof items. Fire in a confined space, such as on board a submarine, could be worsened by the toxic products of combustion from some of the plastics and batteries found in COTS IT. However, there is convergence as commercial environmental specifications toughen and military ones relax.

Qualification

Qualification may be as fit for purpose, mission critical or safety critical. This requires testing to prove usability, environmental survival, reliability, maintainability and types of failure mode.

Traceability

While military equipment is normally traceable, in the sense that each part and each work package is carefully referenced, by and large such traceability does no exist in COTS IT products. Thus, where safety is an issue, it is difficult to provide proven safety cases.

It is noticeable, however, that certain industries are now converging on this military requirement and demanding traceability from the component suppliers and sub-contractors. Typical is the vehicle industry, which needs to know which particular vehicles to recall for safety checks. Much of this change is being driven by litigation concerns and is likely to apply increasingly to COTS IT, particularly hardware.

Reliability

COTS software is notoriously unreliable and prone to regular crashes, although it may well be better than certified custom military software of similar complexity. There is no database of failures and no traceable records for COTS IT, though the wide user base of much software does provide a degree of confidence. At the same time, there are problems with product liability and virtually valueless warranties for software. As mentioned earlier, the US DoD does prove COTS IT by both board and equipment level testing.

Acceptable MTBF and MTTR are key issues normally considered in the very early stages of the procurement cycle. The actual figures required will depend on a number of issues including the operational role, the operational environment, the consequences of failures and the ease of maintenance.

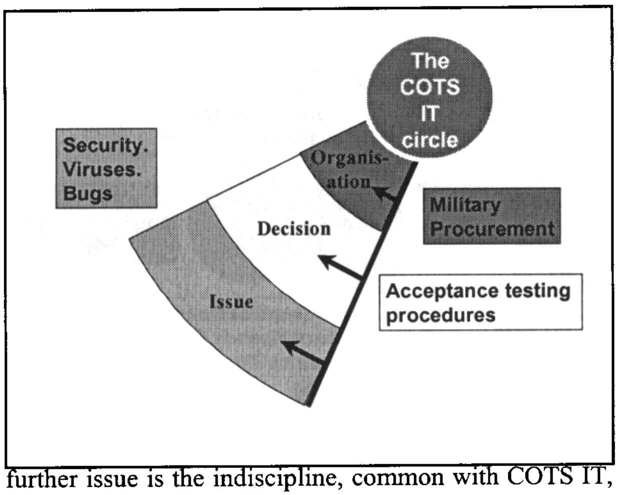
Security

The requirements of military security are currently different from commercial requirements, though the latter are being driven by the need for financial transaction security. Unfortunately, interfacing military crypto requires modification to standard COTS software and re-modification each time the software is upgraded.

The greatest problem lies with transmissions to and from platforms, where radio links are essential and can be intercepted. Commercial crypto usually takes several years to get accredited for military use and is, of course, also available to potential enemies.

Hackability

Designers leave built in trap doors in their software to allow future access. Hackers familiar with COTS software may readily exploit these entry points. A further issue is the indiscipline, common with COTS IT, in the use of passwords, a trend that has been accelerating with the increasing number of passwords and pin numbers that each individual has to remember.



Viruses

One or more viruses may already be resident in COTS IT software, while world familiarity eases its infection and requires care to avoid providing entry points for viruses during the life of the equipment. In general, viruses written to work in custom military software are only likely to be generated by professionals employed by potentially hostile nations.

Bugs

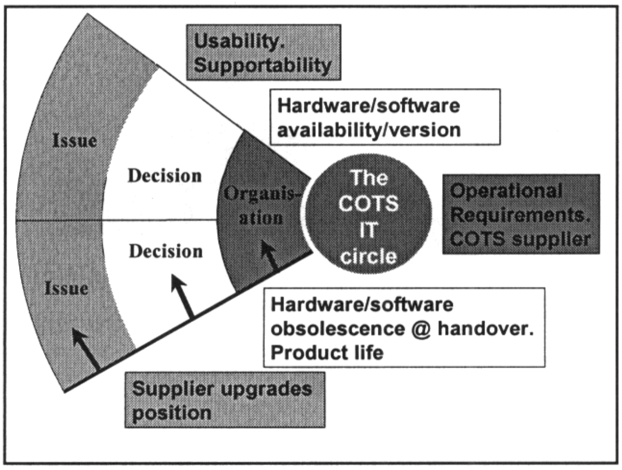
All software contains bugs. These may occur at different frequencies and with different impacts on the user. There are bugs that may occur daily, weekly, monthly, yearly, once a decade or even once in an equipment's lifetime. One of the major difficulties is testing for bugs and in this area, COTS IT fairs well with large numbers of Beta testers, not to mention the often-large installed base. However, removing bugs has the unfortunate habit of introducing new ones, so that it may well be better to live with a number of bugs if their consequences are not severe.

Supplier upgrade position

No COTS IT supplier can continue to offer a standard product in the market place for very long without upgrading it. There are a number of drivers for this approach:

- Competition from other suppliers.
- Inadequacies in the existing product.
- The need to broaden the capabilities of the existing product.
- For software suppliers, to take advantage of improvements in hardware speed and memory.
- For hardware suppliers, to benefit from improvements in component and sub-system technology.

The result is that improved hardware comes on the market in a matter of months, while upgrades to software appear every one to two years.


Usability. Supportability
Hardware/software availability/version
Issue
Decision
Organis-ation
The COTS IT circle
Operational Requirements. COTS supplier
Decision
Issue
Hardware/software obsolescence @ handover. Product life
Supplier upgrades position

Usability and supportability

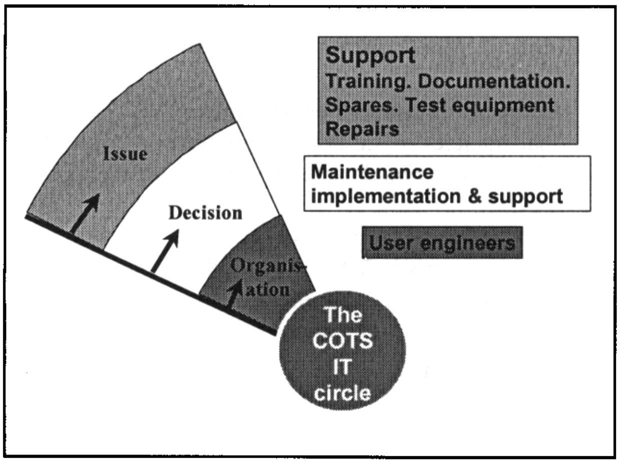
The general familiarity of people with COTS IT helps usability as does competition in the market place. Supportability usually depends primarily on the degree of obsolescence when the equipment is handed over to the user. Because of short COTS IT product lives, supportability will often depend on the prime contractor leaving the actual choice of COTS hardware and software as late as possible in the delivery programme.

Rapid obsolescence

Two years seems to be the time span before the issue of a major software upgrade for any particular program. From that point on, the older version ceases to be supported. The appearance of new hardware models is measured in months rather than years. Thus, the use of COTS IT forces the upgrade route on the purchaser when the supplier ceases support. Consideration should be given to upgrading only that hardware necessary to support new software needs.

Support

The term support is used here as the activities needed to enable equipment to be kept available for operational use. It includes training and provision of documentation for users and maintenance staff, as well as the supply of spares and test equipment.


Support
Training. Documentation. Spares. Test equipment Repairs
Issue
Maintenance implementation & support
Decision
User engineers
Organis-ation
The COTS IT circle

Training

On the whole, COTS IT suppliers do not provide training in their products. This task is largely left to established training companies and can sensibly be incorporated in the equipment prime contract.

Documentation

COTS IT documentation is very thin, the choice of all suppliers being to provide the vast bulk of the information on CD ROMs. This format may change as DVDs and other new media replace CD ROMs. Some software programs also provide on-line support for registered users, allowing them to downloaded updates from supplier web sites. All this help may be printed out, but it is in the form of very basic word processing pages, largely without illustrations, rather than the excellent standard of most custom (and hideously expensive) military handbooks.

Spares and test equipment

Much COTS IT hardware is not designed for repair and thus spares support is limited both in extent and duration. In the commercial market, failed hardware is normally discarded if it fails outside the extended warranty period – normally three years. In any case, manufacturers' warranty repairs are likely to be of a form fit and function nature, where the failed unit may simply be replaced rather than repaired.

Much COTS IT hardware includes diagnostic software to facilitate faultfinding.

Repairs

The support of COTS IT demands a different maintenance policy to that in place for existing military equipment. COTS IT hardware is relatively reliable and, in the event of a failure, much of it is designed to be thrown away rather than repaired. Spares at board level are available only for a short period for current products and are not NATO codified. Both repair work and IT training have been largely sub-contracted by the IT industry. Whether, in these circumstances, prime contractors can provide long-term logistic support remains to be seen.

Modifications and half life updates

Modifications

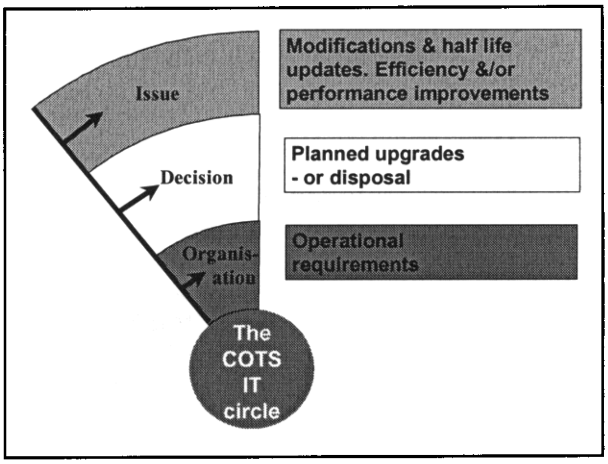
Contractors are very reluctant to undertake modification to in-service COTS IT, and government financiers are somewhat reluctant to accept the standard upgrade process. There is a need for transparent interfaces and architecture at the start of any COTS IT-based programme to support future growth. Furthermore, any modification may introduce new bugs.

Performance and efficiency improvements

There are two main purposes for using any IT. The first is to improve ways of undertaking tasks to provide improved performance. The second is to do the task more efficiently. The fact that the equipment contains COTS rather than custom IT should not reflect on this issue. Efficiency improvements can be obtained by using, for example, a standard COTS human-machine interface to avoid the need for retraining operators before they move to a new role.

Disposal

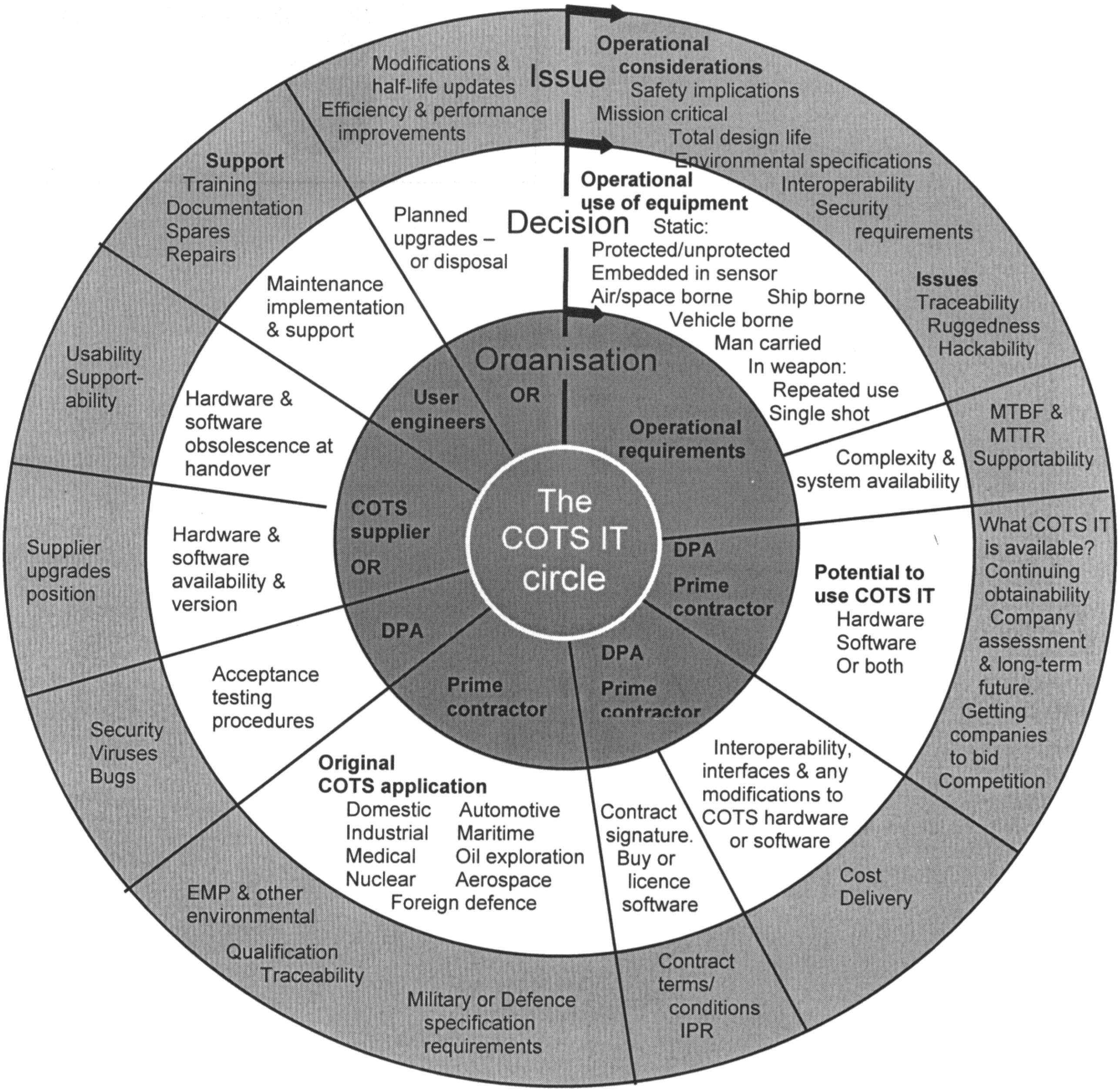
The decision to dispose of a piece of COTS IT is generally straightforward. However, care must be taken to ensure that any classified data are entirely and effectively removed from any storage medium, such as floppy and hard disks, CDs and tapes. Special care must then be taken with their declassification or destruction.



The COTS IT Circle helps identify issues, the decision that must be made and who has to make them, when using COTS IT in military equipment

Abbreviations
MTBF – Mean time between failures
MTTR – Mean time to repair
IPR – Intellectual property rights

DPA – Defence Procurement Agency
OR – Operational Requirements



Standards – Myths, Delusions and Opportunities

(February 2000)

Nic Peeling
(DERA Fellow)
Richard Taylor
(DERA Senior Analyst)

Defence and Evaluation Research Agency,
Woodward Building, DERA Malvern
St Andrews Road, Malvern
Worcs., WR14 3PS, UK
N.Peeling@eris.dera.gov.uk
R.Taylor@eris.dera.gov.uk

Introduction

This paper describes how a new approach to defence standardisation could deliver, for the first time, the benefits that defence standards and Open Systems have for so long promised.

The paper traces the history of defence computing standards. It examines the original benefits that standardisation promised in the defence arena. It examines why so many defence standardisation efforts have failed to deliver on those promises. It then goes on to examine why the original efforts to create a standards-based computing market (the Open Systems movement) also failed. The limitations of a standards-based approach will be described from both a technical and commercial viewpoint. The paper concludes with an optimistic message, that the Internet Standards and the Open Source movement have the potential to deliver on the original promise of the Open Systems movement.

Original benefits promised by standardisation

In the UK, computing standard efforts started in the mid 1960s with the standardisation of Coral 66 as the standard high level language for real-time software, and the Ferranti Argus M700 as a standard computer architecture. The prime benefit intended for such standardisation was the reduction in through-life maintenance costs for software and hardware by reducing the diversity of programming languages and computers utilised in UK MOD systems.

A subsidiary benefit of these early standardisation efforts was the increased portability and reusability of software written in Coral 66 and Argus M700 assembler.

Coral 66 was invented because no existing commercial language (such as Algol 60) had the necessary list of mandatory features:

- Deterministic behaviour needed for real-time, embedded applications;
- Highly efficient run-time code;
- Support for structured programming.

Reduced diversity and *application portability* have remained two of the enduring benefits sought by defence standards.

In the 70s and 80s additional benefits were pursued by defence standardisation efforts:

- Promotion of best practice to industry (e.g. PCTE);
- Interoperability (e.g. ISO OSI);
- Promotion of a market in competing, but compatible, implementations (e.g. Ada and PCTE).

Why did the original promise so often fail?

Although Coral 66 is remembered with some affection, most defence standardisation efforts have either failed totally (e.g. PCTE+), or have been abandoned after the mainstream market passed them by (Ada), or have locked the defence community into niche products (ISO OSI's X.400). The principal reasons for this limited success are:

- You cannot buck the market (e.g. Ada and ISO OSI); eventually COTS products make defence-specific niche products look too expensive, with too little product support;
- You can never truly create a homogeneous defence world (e.g. a country still has to interoperate with its allies, and its suppliers);
- Standards created by committee are often either "lowest common denominator" or very difficult to implement. This leads to industry de-facto standards shooting ahead (e.g. TCP/IP).

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

The Open Systems market

By the early 80s the lack of success of defence standardisation efforts was widely understood, if not openly acknowledged. It was at this time that the UNIX supply industry coined the term Open Systems and standards organisation such as IEEE (with POSIX), the OSF and X/Open rose rapidly to positions of great prominence. The defence community saw the Open Systems movement as a chance to reduce operating system diversity, enabling application portability, allowing competitive hardware procurement, all within a framework that commanded mainstream COTS support. Not surprisingly the defence world were early, enthusiastic supporters of the Open Systems movement, with many countries adopting Open Systems standards within their defence computing policies.

Yet again the defence world had backed a loser. The principal reasons that the Open Systems movement fizzled out were:

- The UNIX vendors could not resist differentiating their UNIX offerings in order to lock customers into their particular flavours of UNIX. Consequently the promise of application portability was undermined, and software vendors usually only supported a few of the largest vendors, and many abandoned UNIX altogether for the more homogeneous Microsoft world;
- The operating system that has the most applications wins. Microsoft tied Windows very closely to the PC, whereas the leading UNIX suppliers tied their operating systems to their own proprietary hardware. As PC sales took off, Windows came to be the favoured desktop operating system for software vendors to support. UNIX and Open Systems retreated into the server operating system market, and in the 90s Microsoft started to take that away from them with their NT operating system.

Common Operating Environments

In the late 90s the UK's MOD accepted that the Open Systems movement was not going to deliver an answer to its needs for computing standards and started the development of Common Operating Environments (COEs) and the UK Defence Interoperability Environment (DIE). The COEs and DIE were comprised of a pragmatic mixture of de-jure and de-facto standards, and proprietary products. Unlike the US's DII COE, the UK approach was standards-based and was not a software build and system integration infrastructure. Consequently the DIE and COEs were intended to promote, rather than guarantee, interoperability and application portability.

The COE and DIE initiatives have promoted a major shift in the procurement patterns of MOD projects, with greater adoption of Windows on the desktop, and a move towards a domain-based approach to security.
The COEs and DIE approach creates a number of challenges:

- Can the definition of the COEs and UK DIE evolve at a rate that matches the furious pace of change in the marketplace;
- Given that the COEs and UK DIE evolve at a similar rate to the IT marketplace, there is a significant issue in either keeping defence systems up to date with the latest COEs and DIE, or of managing multiple legacy systems;
- The situation of whether a pragmatic approach that includes de-facto and proprietary standards is consistent with guidelines for open competition, is not totally clear.

Given that the benefits of the COEs and UK DIE are less clear cut than an approach that seeks to guarantee interoperability and application portability; and that the costs of maintaining and applying a rapidly evolving set of standards will be non-trivial; only time will tell if the COEs and DIE approach is cost effective.

The Way Ahead? - Internet Standards and Open Source

The last two years has seen a phenomenal growth in the usage and profile of both Internet standards (such as HTML and XML) and Open Source implementations (such as Linux). Both these movements have been fuelled by the dramatic growth of the Internet. These movements are driven by forces that make them of particular interest to the defence community:

- The Internet by its nature is not tied to any particular proprietary hardware or software platforms;
- The Internet's focus is on interoperability. This coincides with the emergence of extranets, which have convinced many organisations that interoperability with the outside world (customers, suppliers and partners) is a more important business driver than intra-organisational interoperability. As a consequence the role of proprietary standards, such as Microsoft Office formats, as a mechanism for interoperability between organisations is in decline;
- The Internet and Open Source communities are led by engineers. This has two very important effects: firstly, that it is relatively free from commercial politics and "dumbing down"; and secondly, that this world takes implementation issues very seriously;
- The Open Source method of licensing software means that it is virtually impossible for manufacturers to produce differentiated products that undermine application portability. For this reason it is possible that Linux may soon become the software vendors non-Windows platform-of-choice;

- The Internet and Open Source communities are able to attract massive development resources, much larger than even a company of Microsoft's size can deploy;
- Open review of source code leads to two very important properties: firstly, open source software over time becomes extremely robust; and secondly, open review is coming to be seen as the key to controlling software vulnerabilities, and the Open Source model makes patches to vulnerabilities available very fast indeed.

The defence world should consider whether the Internet and Open Source communities are now delivering on the promise of the Open Systems movement. In addition there are benefits offered that go beyond anything that current standards can provide:

- Open Source may be the only way of getting the twin benefits of COTS support and visibility of vulnerabilities;
- Open Source may offer an alternative to GOTS and niche-COTS solutions to defence-specific requirements;
- It may be possible to develop defence-specific variants of Open Source programs;
- Given the technology focus of the Internet and Open Source communities, it is possible that the defence world can influence the direction of these communities.

Conclusions

Defence standardisation efforts have traditionally been frustrated by the rapid rise of de-facto COTS standards.

The latest UK defence standardisation efforts based on COEs and the UK DIE are based on a pragmantic choice of de-facto, de-jure and proprietary standards. Only time will tell if these latest efforts provide the benefits of standardisation in a cost-effective way which can keep pace with the rapid developments in the IT marketplace.

This paper argues that the defence community should consider whether the latest developments in Internet Standards and Open Source, offer an opportunity to capture the benefits of Open Systems which the UNIX industry squandered in the 1980s.

*E*nvironment for *S*ignal *P*rocessing *A*pplication *D*evelopment and Pr*O*totypi*N*g - ESPADON

Bob Madahar, Jan Hunink[1], Gilbert Edelin[2], James Smith[3], Brigitte Saget[4]

BAE SYSTEMS Research Centre
West Hanningfield Rd,
Gt. Baddow, Chelmsford CM2 8HN, U.K.
bob.madahar@gecm.com

Abstract: Defence industries are increasingly expected to field state-of-the-art products, at significantly lower costs, over significantly shorter time scales, and with significantly greater functionality. New designs, as well as design upgrades, are expected to keep pace with technology advancements, particularly in microelectronics. These constraints, and others, are forcing industry increasingly towards Commercial Off The Shelf (COTS) components (hardware and software). The advantages are reduced costs and state-of-the-art technology compared to proprietary in-house developments, and hard-wired solutions, which have long development times and are invariably out of date by the time the product is commissioned. The disadvantages are principally non-compliance with rigid military specifications of the COTS components and the inability of defence industry product design development and integration methodologies, established over many years, to accommodate the COTS components in an efficient and timely manner. Obsolescence (more acute for bespoke designs) created by COTS components for the long life-cycle military products, is also a key concern and leads to costly retrofits unless the potential design upgrade is included in the design methodology.

These major concerns are being addressed for defence embedded signal processing applications by the tri-national European EUCLID/Eurofinder defence programme called ESPADON. The primary objective is to significantly improve (reduced cost and timescales) the process, by which complex military digital processing systems are designed, developed and supported. A new design methodology, and a new development environment, has been reinvented to support this aim through reuse, concurrent engineering, rapid insertion of COTS technology and the key concepts of rapid and virtual prototyping. These techniques and developments are presented in this paper.

Keywords: ESPADON, Methodology, Prototyping, DSP, COTS

1 INTRODUCTION

Over the last decade there has been a sea change in the climate for the development of military digital processing systems. Principal factors forcing the change are:

The New World Order – the end of the cold war has seen a dramatic reduction in the defence budgets worldwide and changed the perceived future requirements. Political changes, economic globalisation, and technology advancements, sometimes on the back of 'local' conflicts, have also brought additional competitors (Israel, South Africa, India,..) into the market place.

The Microelectronics Revolution – the exponential growth in the performance of microprocessors and associated electronics (> 10M transistors/device and rising, Memory x 4 every 3 years [>256Mbit DRAM, >8Mbit SRAM], Clock rates x 50 every decade). This is continuing apace (1.5 order of magnitude increase in performance every decade, Moore's law survives!) with no immediate signs of abating or hitting the fundamental limits of physics (see Fig. 1)

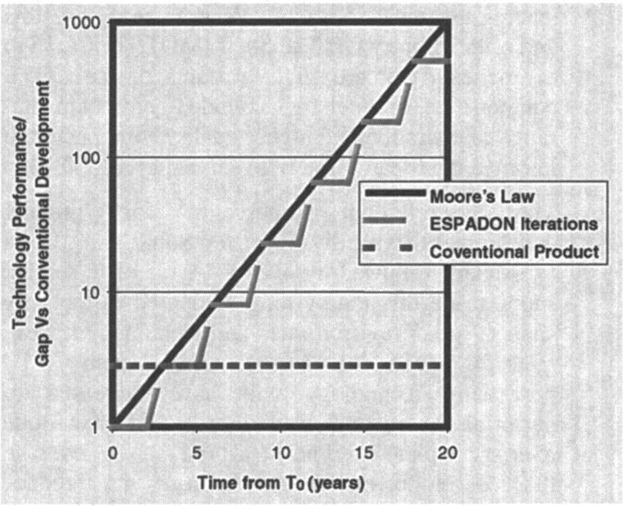


Fig 1. Iterative Development Methodology

[1] Signaal, Zuidelijke Havenweg 40, P.O. Box 42, 7550 GD Hengelo, Netherlands
[2] Thomson-CSF Detexis, 1 Bld Jean Moulin, 78852 Elancourt CEDEX France
[3] Thomson Marconi Sonar Ltd., Dolphin House, Ashurst Drive, Bird Hall Lane, Cheadle Heath, Stockport, Cheshire, SK3 0XB, U.K.
[4] Matra BAe Dynamics, 20-22 rue Grange, Dame Rose, B.P. 150, 78141 Velizy-Villacoublay, CEDEX France

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

The New Age User – the customer/user expectations are ever more demanding in terms of system inter-operability, functionality, capability, and cost (better, faster, cheaper). Note this is conditioned, perhaps unfairly, by their daily exposure to the high performance, fast graphics, highly integrated and easily networked PC environment available on their desk top.

These conditions give rise to the adoption of COTS and the demise of the conventional military methodologies and bespoke military specific developments of signal processing systems for the following reasons:

a) For a specific application, at a given time, the optimum (performance) signal processing designs are likely to be bespoke non-standard hardware and interfaces, software optimised for the specific hardware, and a performance driven unique solution. Such 'company-centric' proprietary developments, and hard-wired solutions, have long development times and are invariably out of date by the time the product is commissioned. This is costly (time and money), the systems are difficult to reuse, and in any case quickly (a few years) overtaken by COTS technology, and emerging standards, and rendered obsolete (technology and components).

b) The support for military specific components by microelectronics vendors is declining, as they position for the significantly larger consumer market, thereby accelerating obsolescence problems and increasing the costs of military specific designs. The COTS components offer significantly better cost/performance ratios that the defence industry must try to adopt to remain competitive and offer a leading technical solution. Vendors recognise this and offer Military Off The Shelf (MOTS or COTS+) components that are slight variants of the COTS components to include extended environmental range of operation and higher quality components to improve the Mean Time Between Failure (MTBF).

c) The conventional timescales for product development (typically 4 yrs for Sonar and 10 yrs for Radar), followed by over a decade of in-service support, are disparately long compared to the very short (~ year) revision rates and technology refresh rates for COTS digital processors. Hence COTS processing technology will have increased in performance by one or two orders of magnitude over the typical lifetime of a product. To leverage these developments, defence industry must reduce development timescales and design for the rapid insertion of emerging COTS technology (design for upgrade) so as to maintain a leading solution for the customer.

d) The significant disparity in timescales discussed above presents defence industry systems with an acute obsolescence problem that is occurring earlier and earlier in the overall product lifecycle. At present there are two methods available to handle such a problem. The first is a lifetime buy and storage of components that are going to become obsolete. This requires capital outlay upfront, hence depreciating in value, based on exiting sales and estimated sales in the future. The latter may not materialise (capital loss) because the system by definition is obsolete compared to the current competing products. The second is an equipment retrofit with the current component technology. Unfortunately, design for upgrade is not integral to conventional military designs. Hence the retrofit is comparable in cost to the initial development and therefore competes unfavourably with current competing products. The solution is a new methodology where the development times are reduced, with design for reuse and design for frequent upgrades as an integral part of the process.

The new methodology, to ameliorate the above concerns, is being developed by the tri-national European EUCLID/Eurofinder defence programme called ESPADON [1] [2] as it is beyond the resources of a single company and Nation. The international consortium comprises Thomson-CSF and Matra BAe Dynamics from France, Signaal from Netherlands, Thomson Marconi Sonar Ltd and the Marconi Research Centre from the United Kingdom. The 3 year duration project, jointly funded by the consortium and by the Ministries of Defence of France, United Kingdom and the Netherlands, started in July' 98.

2 NEW METHODOLOGY & TECHNIQUES

ESPADON is the European analogue, albeit with only a few % of the budget, to the U.S. tri-service research programme RASSP (Rapid Prototyping of Application Specific Signal Processors) which was initiated in 1993 with a budget of $150M and lasted for nearly 5 years [3]. RASSP was a very broad programme involving government, defence industries, Electronic Design Automation (EDA) industries and Academia investigating three principle threads – Architecture, Methodology and the Education and Enterprise Infrastructure. The focus of ESPADON however is much narrower, towards the methodology and environment for embedded signal processing applications, and benefits from the lessons learnt by the RASSP programme.

The main project thrusts are:

1) Synthesis of an advanced design methodology and processes for the development of the next generation real-time signal processing systems;

2) Analysis and evaluation of COTS tools, emerging standards, signal processing and communications libraries, and associated techniques of direct relevance to the methodology;

3) Implementation of an ESPADON design environment (EDE), based on the integration of best of class COTS tools, standards and techniques,

within an extensible software framework, that can support the methodology;

4) Demonstration of the objectives through the implementation of real-time adaptive signal processors for Radar and Sonar applications on COTS hardware platforms;
5) Measurement of metrics to quantify the productivity gains and to validate the EDE, the techniques, and the methodology; and,
6) Dissemination of the project and results via the internet, seminar and workshops aimed at European companies.

Of these, the progress midway through the project is that, the methodology has been specified (1), the initial set of COTS tools for the EDE selected (2), and a preliminary version of the EDE implemented (3) and the benchmarking activity begun (5) as described in the sections below.

2.1 Methodology

The conventional methodology for signal processing system or component development is analogous to that for software engineering in the late 70's. It can be represented by the sequence of different activity steps, Requirements-Specifications-Design-Implementation-Testing, where a new step begins when a previous one has ended. The sequence is known as a 'waterfall', or with iteration to previous steps as an 'iterative waterfall' or the V model where hardware and software are co-developed for a system, Fig 2. These methods have been shown to be deficient for software engineering [4]. They fail to recognise the role of iterations in the overall process and the specifications are frozen at an early stage of the development process. The implication of the latter is that the cost committed to the program is large before the system concept has been adequately proven in terms of risk and performance. Iterations are to reduce risks, verify – are we building the product right?, validate – are we building the right product?, and test the outputs of each activity before proceeding to the next or to a previous activity to take corrective actions. Failure to do this results in validation late in the development process by which time corrective actions are costly as they propagate backwards through the process. For these reasons new methods have been developed for software engineering, and applied successfully, but have not as yet been applied to signal processing. A key method is the risk driven spiral model where risks are analysed, versus key criteria, at each step and the developments refined through successive iterations to eventually converge to the final solution [5].

2.1.1 The Iterative Development Process

ESPADON has, after careful analysis, adopted and modified this method and defined a new methodology for signal processing application development. This is shown in abstract form in Fig. 3 and applies to the development of the signal processing subsystem at the highest level. Central to the new methodology are the following key processes:

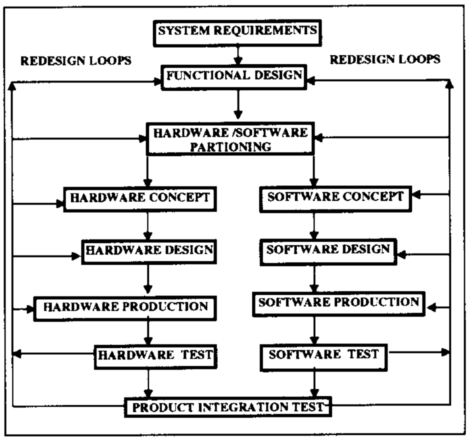


Fig. 2. V Model of Development

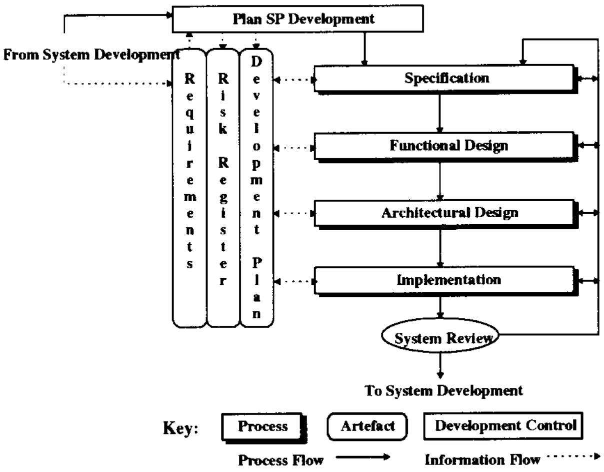


Fig. 3 Abstract Iterative Development Lifecycle

Specification – refinement of the raw requirements from the system development into an engineering specification that includes salient functionality, interfaces, physical attributes and performance and cost criteria.

Functional Design – the functional parts of the component specifications are modelled by assigning the appropriate algorithmic and control processing blocks, functional libraries and description models of computation, to a functional model. The model is independent of the implementation and is simulated to prove functional correctness or raise any corrective actions for further refinement of the overall process.

Architectural Design - The critical characteristics of the reference functional model (computing power, rate, etc.) and the non-functional requirements (costs, volume, etc.) are identified. A risk analysis is performed to determine the critical characteristics to be taken into account in identifying candidate architectures. Through trade off studies, the most effective architecture is chosen. If no appropriate solution can be found, the model and/or the system requirements are refined.

Implementation - the result of the current design iteration, a Rapid Prototype, a Virtual Prototype (section 2.1.3) or a Production Component. This process includes production and test of hardware and software, integration of the software on the target hardware and validation of the component. Co-design, Hardware/Software synthesis and co-verification are essential techniques to use in this process.

The other nodes shown in the diagram control these processes and the development lifecycle for the signal processing component being developed. Namely,

Plan Development – input is the requirements and the output is the plan and risk register

System Review (Control Point) – a system level review, with all the system design authorities, at the end of each complete cycle in accordance with the plan. Outputs are, a) exit with the results (the appropriate artefacts) to the overall system development team for integration with the system, or b) reiteration of the cycle with changes to the control artefacts as necessary.

The control artefacts are the *Requirements* – handed down from the overall system design process, *Risk Register* – severity and priority ordered list of current identified risks, and the *development plan.*

Each of the key processes above is itself composed of the generic abstract iterative process shown in Fig. 4 where the nodes either represent generic activities, described below, or the control artefacts described previously. These generic activities are the five phases that embody the ESPADON iterative design methodology:

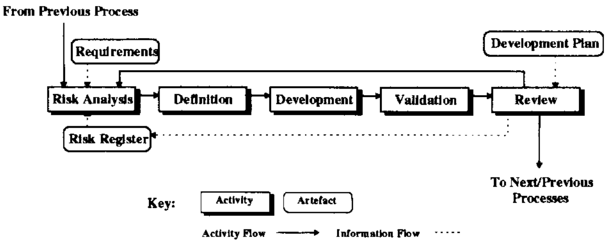


Fig. 4 Anatomy of an Iterative Process

Phase 1: Risk Analysis - analyse the requirements, any available process artefacts, the risk register and the development plan to determine what should be achieved in the current pass through this process.

Phase 2: Definition - the definition and documentation of the objectives for this iteration. This will include creating or updating any design and test documents and/or data involved.

Phase 3: Development - develop the object(s) (one or more of the design artefacts) of this iteration according to the definition made in the previous activity.

Phase 4: Validation - validate the object(s) produced by this iteration against the objectives and the component requirements using the defined tests. Analyse the results and update the risk register.

Phase 5: (Exit OR Refine) Review - review the requirements, any available design artefacts, the risk register and the development plan to determine what course of action needs to be taken next. Possible actions are: a new iteration of the same process (introducing new requirements or refining existing ones) or move onto the next process. If a new iteration of the process is required and this is not compatible with the current development plan, then a Development Review must be initiated and the plan updated.

Design artefacts, not shown in the diagram, will be produced and modified by the activities as the development iterations proceed. As the iterative process proceeds these artefacts will grow in content and become more refined. At the end these artefacts, with the control artefacts, will be the signal processing components complete design archive which can be reused for the development of similar components and mid-life technology updates.

The abstract iterative processes described above are equivalent to a spiral model for signal processing development at the component level, sub-spirals or fractals of the spiral model, or the signal processing development at the system level as shown in Fig. 5. Embedded within this development process are the key ESPADON design concepts defined in the next section.

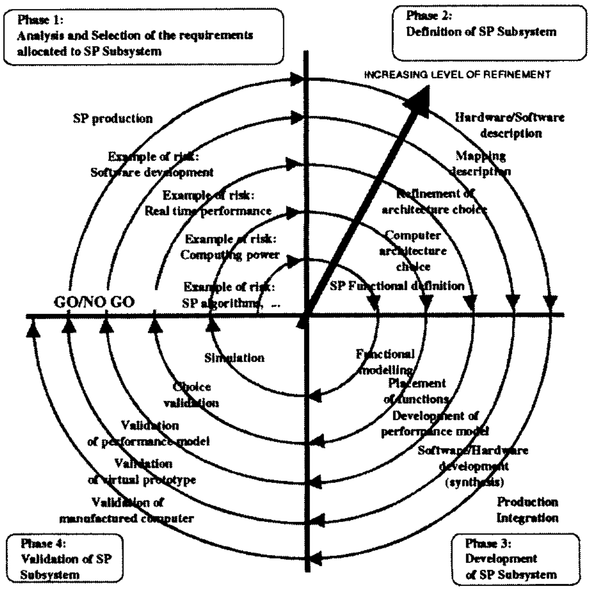

Phase 1:
Analysis and Selection of the requirements allocated to SP Subsystem
Phase 2:
Definition of SP Subsystem
INCREASING LEVEL OF REFINEMENT
SP production
Hardware/Software description
Example of risk: Software development
Mapping description
Example of risk: Real time performance
Refinement of architecture choice
Example of risk: Computing power
Computer architecture choice
Example of risk: SP algorithms, ...
SP Functional definition
GO/NO GO
Simulation
Functional modelling
Choice validation
Placement of functions
Validation of performance model
Development of performance model
Validation of virtual prototype
Software/Hardware development (synthesis)
Validation of manufactured computer
Production Integration
Phase 4:
Validation of SP Subsystem
Phase 3:
Development of SP Subsystem

Fig. 5 Spiral Model of ESPADON Development

2.1.2 Reuse & Capitalisation

Reuse, along side the iterative development process, is the other element of the signal processing methodology implemented to decrease development time and cost. Reuse applies at two levels:

Reuse between iterative processes of development cycle - use elements developed in an iterative process with a certain level of refinement for the development of the next iterative process having a higher level of refinement. The strategy with reference to the generic iterative process is:

Definition activity - the same modelling formalisms or functional models are used at different levels of refinement but with dual libraries of components,

Development activity - hardware is synthesised and code is generated for different target machines with the same synthesis techniques. These targets may be, for example, a workstation or a real time multiprocessor machine according to the development stage,

Validation activity- the virtual prototypes of the previous iterative process are used as a reference for the virtual prototype of the next iterative process.

Reuse of existing components (SP algorithms, components, hardware architectures, PCBs, etc.) – use in-house components already developed or COTS components for the development of an activity (or an iterative process) of the development cycle. The development strategy is:

Development with reuse - development of an application must be able to reuse already-developed existing constituent parts.

Development for reuse (or capitalisation) - the new constituent parts of an application are developed in order to be reused in other systems.

The above reuse objectives are integral to the ESPADON development process and enables,

a) *increasing productivity and decreasing development time,*

b) *providing additional architecture choices,*

c) *using better quality constituent parts since they have already been tested and validated, and*

d) *capitalising on existing know-how.*

2.1.3 Rapid and Virtual Prototyping

An iterative development method necessarily implies the use of prototyping, at some level, such that requirements and functional solutions (the prototypes) can be validated and verified by measurement and improved through successive refinement to arrive at the final solution. The value for complex systems development was recognised in software engineering a few decades ago and high level environments to support prototyping, and faster iterations of prototyping (rapid prototyping), developed [6].

Rapid Prototyping - Unlike software engineering where the functions are compiled and executed to run on a workstation, signal processing requires the functional solutions to be partitioned, mapped and implemented on the embedded multi-processor hardware for meaningful performance measurements and validation. For conventional developments this is a specialised and expensive activity as the code is hand mapped and hand crafted for optimum performance on, as explained earlier, rapidly obsolete custom computers. Instead we need a prototyping environment that is fast and can support the insertion of the available commercial technologies based on COTS boards or COTS computers integrated with any necessary proprietary hardware (I/O, display etc.). The rapid prototype will enable the functionality to be properly tested in terms of dependencies, performance and real-time behaviour. This can be applied to any signal processing component development and associated requirements. It provides an opportunity for the early and frequent involvement of the customer to refine the requirements and common understanding of a signal processing component or iterations of the signal processing system. Rapid prototyping for signal processing is therefore the ability to seamlessly move from the functional design to the architectural design *(the modelling & simulation domain)* to the implementation, through automatic code generation, on real-time COTS test beds *(the execution and measurement domain).*

Note that the prototyping is an iterative development process where the results are used to refine the successive iterations as per the generic development processes described earlier. Clearly as the iterative process proceeds, performance and behavioural data are amassed, the functional models grow in content and become more refined. These successive prototypes, together with their associated functional models and performance and behavioural data, provide the basis for virtual prototyping.

Virtual Prototyping - is the ability to model and simulate in *the software domain*, the complete signal processing application, including hardware at different levels of abstraction, to validate the architecture selection prior to technology implementation. Rapid prototype measurements and information feeds into the virtual prototype, which enables component and system libraries and data bases to be built so as to construct virtual models of signal processing systems for iterative cost/performance and other trade off and analysis studies. Integral to the studies is the concept of hardware and software co-design discussed next.

Co-design – this is implied in the prototyping described above but has particular relevance to virtual prototyping as follows. Co-design is defined as the concurrent and co-operative design of information processing sub-systems composed of hardware and software components operating together. It is central to the iterative prototype developments discussed earlier. In the traditional 'V' model (Fig. 2), the hardware and software developments are partitioned early in the development lifecycle. Hence they diverge in terms of engineering or design interaction, and cross-validation, until the integration, test and validation phase. This phase however is much further downstream leading to potentially costly (time and money) reengineering of solutions, often in software as the hardware is by then fixed, to overcome deficiencies with respect to the initial requirements and specifications (as per the issues in Section 2.). Co-design provides a method to overcome these deficiencies by closely coupling the hardware and software developments within an iterative design framework. The main phases are shown in Fig. 6. The important points are that the design starts with a system or sub-system *specification and functional model.* This specification may be independent of the future implementation and the partitioning of the system into the hardware and software components. The specification has to be captured in a functional model that can be simulated and verified. This model is *partitioned* into hardware models and software models that make up an overall architectural model of the system (the virtual prototype discussed above). These models will at the lowest level be described in high level languages such as C and VHDL and at the highest level be described by graph based objects.

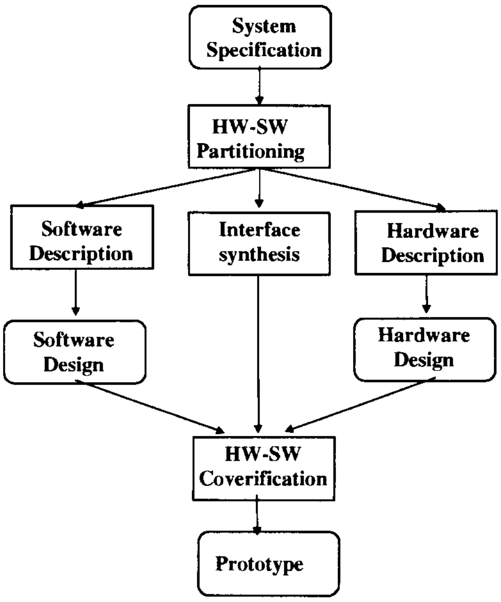


Fig. 6 Co-design Process flow

With the emergence of large reconfigurable and reprogrammable devices (>> Millions of Gates), and system on a chip (SOC) devices, co-design offers a very powerful technique for encapsulating by design software functionality onto hardware devices through partitioning studies and trade-off studies so as to arrive at an optimal architecture. Because it is model based, it is easier to modify and refine the models and architecture for the latest implementation in the succession of prototypes.

Hence the co-design methodology provides the ability to model the system specification, to model the architecture solution and to perform trade-off studies (performance, cost, power consumption etc.). A key application in signal processing application development is in the partitioning and mapping of time and performance critical signal processing functions, that would otherwise run on COTS general purpose processors, onto COTS or bespoke FPGA or SOC arrays.

2.1.4 The Model-Year (MY) approach

The concepts of rapid and virtual prototyping for signal processing are fundamental to the MY architecture concept developed under RASSP [7] and is integral to the ESPADON iterative development methodology. A MY approach expects that the signal processing system can be fielded with the latest digital technology in less than a year if the architecture has been developed iteratively through a succession of prototypes. In fact the MY concept is to deal with obsolescence and provide systems with the latest COTS digital technology when fielded. Key attributes of the MY concept are;

a) the MY mitigates the risks of the development of an equipment by rapidly validating its requirements through a succession of prototypes (Rapid Prototyping); and

b) the implementation of a MY architecture of the signal processing application uses the available digital technology.

Indeed, instead of developing expensive and rapidly obsolete custom computers, the rapid prototype integrates available commercial technologies based on COTS boards or COTS computers. On the other hand, the final equipment, taking into account the constant digital component improvement, is developed with the latest technology. Therefore with the MY architecture, a retrofit of the equipment due to an obsolescence of components is only another iteration in the life cycle of the equipment.

These iterative technology insertions are shown in Fig. 1 as the 'ESPADON technology staircase'. The signal processing system prototypes are refreshed with the latest COTS technology at regular intervals which in practice will be determined by the planned refresh rates for the pre-production, delivery, and post production phases of the signal processing system as part of the iterative development methodology.

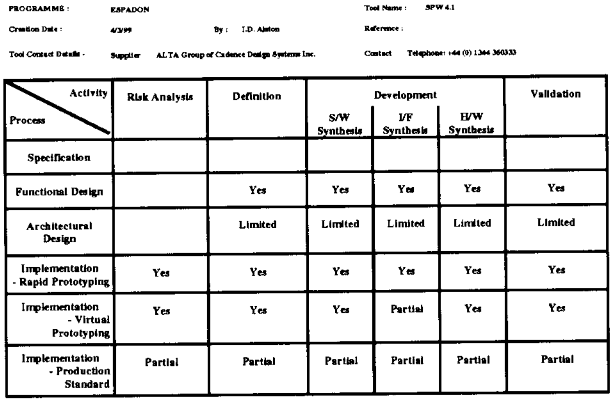
3 THE ESPADON DESIGN ENVIRONMENT (EDE)

Having defined an ESPADON methodology and development process, the next technical development was the ESPADON Design Environment (EDE) to support this new method [8]. Figure 3 described earlier, shows the key development activities which need to be supported by the EDE. For each, the technical requirements, pertinent techniques, and scope was identified and defined. Technical studies were undertaken to provide up to date information on key techniques such as software synthesis, hardware synthesis, rapid and virtual prototyping, libraries, tool interfacing techniques, etc. Each study summarised the current status of the technology areas and potential

COTS tools that were available to support it. These COTS tools were evaluated further as described below.

3.1 COTS Tools Selection

Having identified the potential COTS tools in the domain areas of interest, a tool selection process was defined [8]. As part of this process, and to ensure consistency, a tool function coverage grid, Fig. 7, matching the methodology requirements was designed and used to rank the tools in each domain [9]. Non-functional requirements, not shown, were also added to the assessment. In parallel, a vendor questionnaire, consistent with the grid, was also sent to the vendors for completion and the results used to update the ranking. The key factors that were taken into consideration for the ranking were; *1) Commercial factors (size of company, licence costs, history, support etc.), 2) Features and functionality support, 3) Interface with existing and to future tools and designs, 4) Methodology support, and 5) Usability*



PROGRAMME : ESPADON Tool Name : SPW 4.1

Creation Date : 4/3/99 By : I.D. Alston Reference :

Tool Contact Details - Supplier ALTA Group of Cadence Design Systems Inc. Contact Telephone: +44 (0) 1344 360333

Activity / Process	Risk Analysis	Definition	Development			Validation
			S/W Synthesis	I/F Synthesis	H/W Synthesis	
Specification						
Functional Design		Yes	Yes	Yes	Yes	Yes
Architectural Design		Limited	Limited	Limited	Limited	Limited
Implementation - Rapid Prototyping	Yes	Yes	Yes	Yes	Yes	Yes
Implementation - Virtual Prototyping	Yes	Yes	Yes	Partial	Yes	Yes
Implementation - Production Standard	Partial	Partial	Partial	Partial	Partial	Partial

Fig. 7 Tool Function Coverage Grid

Following the collation and analysis of the results, the best of class tools were selected for detailed evaluations with representative test applications [10]. As the first release of the EDE is directed towards functional design and rapid prototyping, the detailed evaluation stage has concentrated on these domains only at this particular stage in the project. The selection process for co-design, co-simulation and virtual prototyping tools has just commenced.

From the results of the evaluation, the best of class tools for the first release of the EDE, and rapid prototyping are:

- GEDAE [11]Currently technically the best of class tool. It is also the recommended tool from the RASSP programme.
- Ptolemy [12] An extensive research & development software suite covering many domains of signal processing and considered to be the father of signal processing simulation tools. It is research quality software, the output of many students and many years of research at the University of Berkley. It is free open source software available directly from the University.

In addition the following has been selected for mathematical work, algorithm development and prototyping.

- Matlab/Simulink [13] Matlab is widely used among project partners

Other than the above tools, the evaluation studies also recommended the use of signal processing libraries and standards and associated APIs, for example for algorithms and communications, to support reuse and capitalisation and provide tool independence for the future. ESPADON has therefore evaluated the following standards;

Vector Signal Image Processing Library (VSIPL) [14] – This standard is being developed by representatives from Industry, with representation from ESPADON, and academia with the goals to:

- Catalyze the formation of an Industry Standard Working Group for Vector/Signal/Image Processing Libraries.
- Create a widely (industry) supported standard API/library for vector/signal/image processing primitives.
- API/Library for single processor and parallel version.
- Foster standardization for sensor software portability such as reuse, interoperability, low cost COTS upgrade path, lower life cycle costs, etc.

ESPADON is adopting the VSIPL API, and investigating the efficient implementation of the VSIPL standard on the ESPADON target platforms and future evolutions, so as to enable reuse and capitalisation of the algorithm developments. These developments are focussed towards application domain libraries, such as for Radar and Sonar.

A draft of the VSIPL standard has been written and has been distributed for final comments and approvals by the VSIPL core members.

Message Passing Interface Real-Time (MPI-RT) [15] – Inter-process communications (IPC) are the glue that binds processing in the ubiquitous multi-processor embedded signal processing systems. An IPC standard offers the potential for code portability, and hence reuse. MPI-RT is such a standard, and like VSIPL, is being developed by representatives from Industry and academia.

MPI-RT is neither a subset nor a superset of MPI or MPI-2 but part of the process to develop a message passing interfaces standard for real-time applications. It has been developed as a middleware API standard to support the real-time paradigms of TIME-DRIVEN, EVENT-DRIVEN (low level and high level), PRIORITY-DRIVEN processing. The Quality Of

Service (QoS) is a key attribute in each case. In fact the delivery of the QoS is central to the MPI-RT philosophy.

The adoption of MPI-RT by ESPADON raises many issues that need to be investigated further. These are concerned with the delivery of the QOS and the efficiency of implementations. Since it is a standard, its implementation is left to the systems or hardware vendors. At present, to the authors knowledge, no such implementations are available for detailed study, except for emulation on a workstation, though most of the major vendors do have MPI-RT in their future road maps. Hence ESPADON is keeping a watching brief on vendor's developments and investigating how the MPI-RT API may be implemented within ESPADON to provide a possible interface to future implementations. A draft standard for MPI-RT has been issued and is available on the web [15].

The other tools required for the EDE are more concerned with the infrastructure, requirements, cost estimation, EDE control and configuration management. The final list of tools selected for the whole EDE are shown in Fig. 8. These additional tools are not critical to the success of ESPADON but need to be interfaced to the EDE to support the overall signal processing application development lifecycle.

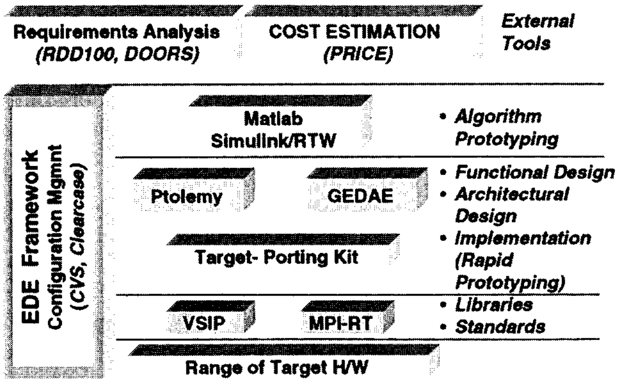


Fig. 8 EDE Tools Selected

3.2 The EDE Framework

The ESPADON Design Environment (EDE) consists of the tools and libraries connected through the EDE infrastructure as discussed above [16]. A technical working group has been established on the project to progress the EDE development through the various revisions, starting with 0.1, the first realisation of a Rapid Prototyping framework, 0.2, the first update, after a 'hands on' evaluation and review by benchmarking teams, and 1.0, the first realisation incorporating Virtual Prototyping. The requirements placed on the EDE are that it is based on :

- a modular approach consisting of standard interfaces etc., an open architecture, and basic services (e.g. key elements).
- existing capabilities/tools: e.g. virtual prototyping tools provide capabilities of co-design, co-simulation and co-synthesis,
- low intrusion into existing tools

It should be portable and extensible and provide functionality that can support the following key attributes:

Simplify tool usage - the new user should have a gentle learning curve

Familiar GUI (rather than command line); On line manual pages – tool selection, usage and style guide; Common tool start up procedures including user profiles.

Make tools appear more 'professional' - some tools have an academic/research origin

Login security, tool usage trace logging; Data security, backup, archive; Design deposition - change control, code management; IP data/design repository, capitalisation and reuse; Multi–user support.

Multi tool management and data exchange - some difficult problems will require the use of several tools at once

Concurrent use of tools for co-design and co-simulation; Sequential use of tools - avoid manual re-entry of intermediate design data.

Tool automation – in some cases existing UNIX or NT tools can be used but they may need a wrapper

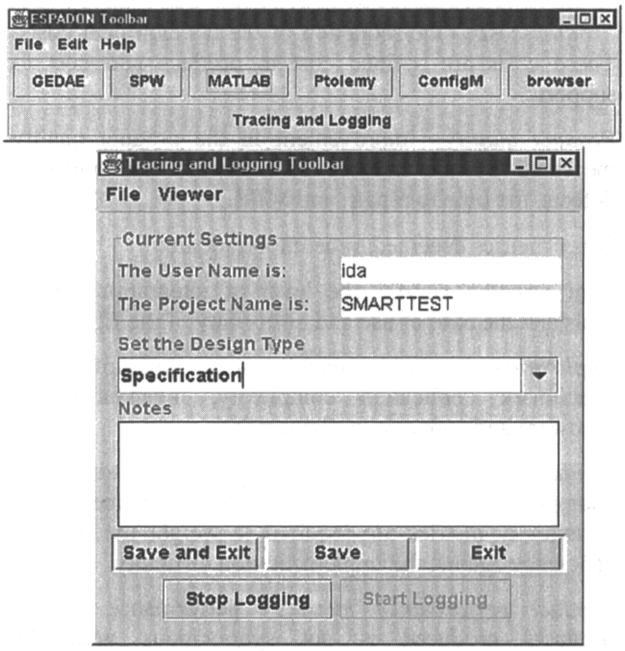
'Scripting' language for driving low level tools.

Other than these attributes, the EDE has to clearly support the iterative system development methodology of ESPADON. Consequently there are three specific viewpoints (users) which govern how the functionality of the framework is accessed and by whom. These are self-explanatory, strictly hierarchical, and are the System Viewpoint (the overall signal processing application composed of a number of component developments assigned to particular project groups), the Project Viewpoint (the signal processing component developments being undertaken by a project group), and finally the User Viewpoint (one of the project group members undertaking a specific task).

To support the above the key elements, *Graphical User Interface, On-line guide, Tool Management & Control, Repository, Data Exchange, and Trace Information* were identified and designed to build the first version of the EDE . The GUI is shown in Fig. 9 with the trace window that records information useful for collecting metrics and the history of the development.

3.3 V0.1 Version of the EDE

This version has been integrated with the GEDAE tool and the Ptolemy Tool and will be used for the benchmarking of an example Sonar application and example Radar benchmarking respectively [17]. The choice of the two tools is deliberate so as to provide performance and efficiency measurements for cross-comparison and mutual improvements. The main objective however is to benchmark the V0.1 version for rapid prototyping by using two representative applications.



ESPADON Toolbar

File Edit Help

GEDAE | SPW | MATLAB | Ptolemy | ConfigM | browser

Tracing and Logging

Tracing and Logging Toolbar

File Viewer

Current Settings

The User Name is: ida

The Project Name is: SMARTTEST

Set the Design Type

Specification

Notes

Save and Exit | Save | Exit

Stop Logging | Start Logging

Fig. 9 EDE User Interface

The Ptolemy tool is being ported to a Mercury platform for the benchmark. The GEDAE tool supports a number of target platforms (Mercury, Ixthos, Sky etc.) but is being ported to support a subset of the EUROPRO platform [18]. Key to both is the board support package for the target architecture, its adaptability to support other targets and the overall efficiency. Hence work is underway with GEDAE to support commonly used real-time operating systems such that additional processors (RISCs and DSPs) can be supported. This will enable a board porting kit to be developed to support a range of hardware test beds and potentially heterogeneous systems. An example of the design flow with GEDAE is shown in Fig. 10.

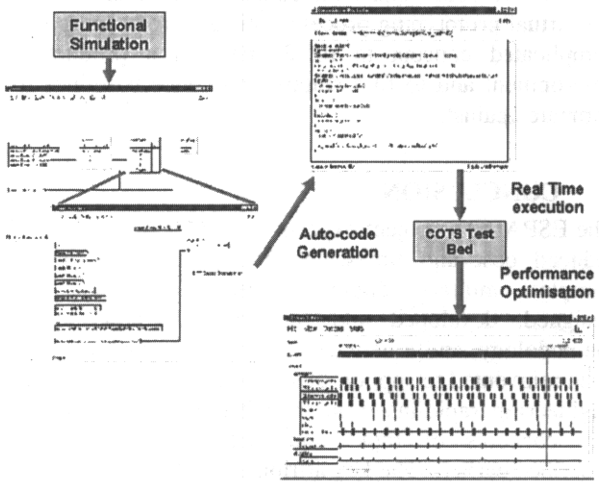


Fig. 10 Typical Rapid Prototyping Design Flow

The first application of the EDE is towards the benchmarking of the ESPADON methodology and development process. Two benchmarks were identified at the outset of the project. These are the implementation of a beam-former for a Sonar and a Radar applications (see Fig. 11). Beamformimg is a generic processing function for which metrics for conventional developments are known or can be estimated. A technical document defining the rationale for, and the definition of the metrics to measure the ESPADON objectives has been written and a benchmarking plan drawn [19]. An overview of the radar benchmarking application is provided in the next section.

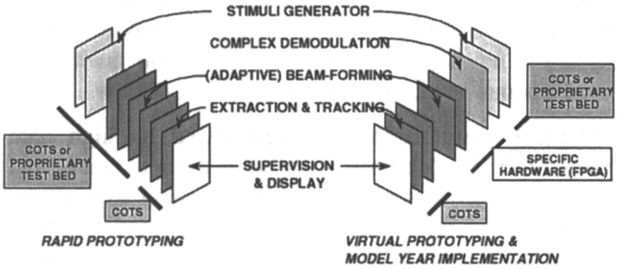


Fig. 11 Outline of the beamformer benchmark.

3.3.1 The Radar benchmarking application [20]

For the Radar benchmark, an adaptive digital Beamformer (BF) application for multibeam radar, Fig. 12, will be used.

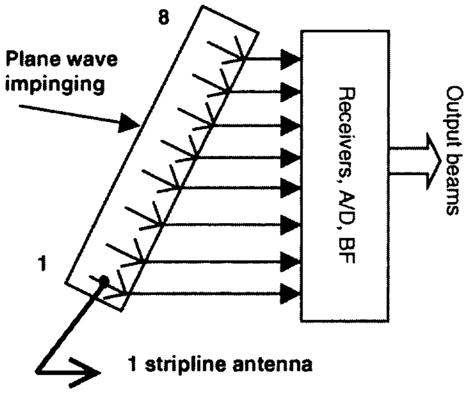


Figure 12: Multi stripline receiver antenna array signals are transformed into a beam pattern in elevation

The function beamformer is part of the functional chain of an X-(H new NATO) band air surveillance radar. The antenna of the radar comprises a vertical array of, for example, 8 elements each of which is a horizontal linear stripline array of dipoles. The array is used as a transmit antenna as well as a receive antenna. As a transmit antenna the power splitter distributes the RF power among the elements (linear arrays) via phase shifters and circulators. This results in a transmit beam which illuminates targets within the desired elevation coverage envelope. As a receive antenna, each of the 8 array elements is connected directly to an individual receiver and an A/D converter. Each array element is sensitive over the desired elevation coverage. Elevation beams are formed by the digital beamformer that performs an 8 point FFT or FIR algorithm on the outputs of the 8 receiver channels. In this way a multibeam receive system is formed, Fig. 13. The benchmark concerns only the receiver beamforming function, the transmit beamforming function is implemented by an analogue system.

The beamformer is adaptive with respect to the ships course and speed, and the ships roll and pitch movement. This results in a phase correction that is applied to the complex data stream prior to the beamforming, together with windowing and calibration correction.

The application contains all aspects of a radar signal-processing element as it is found in modern radar systems nowadays. This includes mode switching, synchronous/asynchronous data and control flow.

Adaptive beamforming is characterised by high data rates (up to 20 Mbytes/sec for each channel/beam) and corner turn processes. The signal processing architecture on which the algorithm will be implemented therefore asks for high-end multi-processor machines with high-speed crossbar interconnect between processing nodes. The selected crossbar interconnect has a peak throughput of 267 MB/s per crossbar connection and also gives the desired flexibility needed for rapid prototyping in the sense of ESPADON. For the processing element the 4th generation Motorola PowerPC processor is selected: the AltiVec processor. This processor is similar to the previous version of the PowerPC with a 128-bit vector-processing unit added, which is well suited for signal processing algorithms.

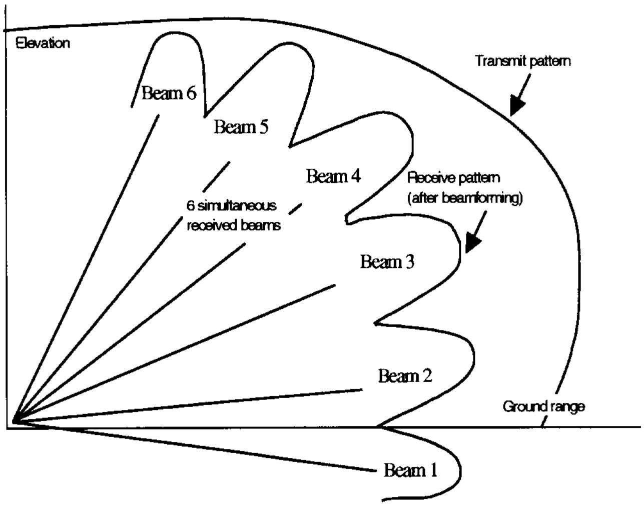


Figure 13: Example of resulting multi beam pattern in elevation for an eight channel to six beam beamformer

3.3.2 Principle Benchmark Metrics

As per the iterative development methodology, the benchmark of the ESPADON process will also be carried over successive iterations. For each of the benchmarks, the principle metrics directly related to the performance of the ESPADON process and performance will be collated. These in summary are:

Design Cycle Metrics – the reduction in development time, through software and hardware reuse, productivity, iterative refinement etc.

Product costs - the reduction of the development costs. These costs are defined cost to produce and cost to support.

Product quality - improvement in the product quality measured by the number of hardware and software defects, the time to repair, and MTBF.

Other metrics deemed to be important are:

Tool oriented metrics -the level of integration of the tools and the ease of use and uniformity of the EDE.

Application complexity metrics – try to capture the benchmark application complexity, independent of hardware and software implementation

Product complexity metrics - for each product, for example, software, hardware, and documentation, complexity metrics are required to weight the product efficiency against the implementation difficulty

Product performance metrics - performance of the products produced is not synonymous with the ESPADON performance itself. Hence it is important that the appropriate metrics are collected and analysed.

These metrics will be collated as part of the benchmarking activity which will be carried out for each of the three releases of the EDE. The first step will be to evaluate the preliminary version of the EDE (V0.1) described above. The results will be fed back to improve the tools, the EDE framework and the integration. These steps will be repeated for the next version of the rapid prototyping EDE and then proceed to the virtual prototyping EDE.

The final release, Virtual Prototyping version, will not be benchmarked against two applications, and by two teams, but against one benchmark application (Radar or Sonar) and one benchmark team. This is expected to be sufficient to demonstrate the concept and advantages of the virtual prototyping process. Virtual prototyping is a complicated concept to disseminate in a production environment and to find suitable reference baselines to compare against.

4 CONCLUSION

The ESPADON project expects to significantly improve reduced cost and timescales, the process, by which complex military digital processing systems are designed, developed and supported. A new design methodology, and a new development environment, has been reinvented to support this through reuse, concurrent engineering, rapid insertion of COTS technology and the key concepts of rapid and virtual prototyping as described earlier. The key attributes of the methodology are a Risk driven spiral lifecycle, encapsulation of the

"Model Year" concept to mitigate risks by the iterative development over successive rapid prototypes integrated with the latest COTS technology, and support for component Reuse and Capitalisation.

The preliminary version of the EDE to support the methodology has been implemented, with the GEDAE COTS tool, and supports the concept of Rapid Prototyping. Key features are the data flow signal processing paradigm, the EDE framework and GUI, the support libraries, and the efficiency of code generators (communications and processing). The first EDE is targeted for a range of real-time COTS test beds, the first being a Mercury system. A benchmarking process to evaluate the EDE and provide valuable feedback towards its improvement has begun. It will enable real behavioural, performance and timing measurements to be made to feedback into the iterative process so as to arrive at an optimum implementation.

Such an EDE and Rapid Prototyping environment provides a number of advantages for signal processing application development. It enables the collection of measurement data to provide as an input to virtual prototyping. The performance data can be used to correctly size the overall system requirements (hardware and software). Data can be collated with respect to benchmarking other COTS components. The prototype can be used with real data or in the field to validate the processing. It enables the early and frequent involvement of the customer so as to adjust requirements over the development and field experimentation stages. These advantages offer a significant improvement compared to the conventional methods for signal processing application development.

5 ACKNOWLEDGEMENTS

The work has been carried out under the WEAG EUCLID/Eurofinder programme, Project RTP2.29, with support from the UK, French and Dutch MoDs and the participating companies. The authors are grateful for this support and would also like to acknowledge the contributions of all the ESPADON team members.

6 REFERENCES

[1] D. Aulagnier and B. Saget, "The ESPADON programme: Environment for Signal Processing Application Development and prOtotypiNg", Actes des Journees Thematiques Universites/Industries du GRAISyHM-AAA-99, Lille, france, 23-24 March, 1999.

[2] D. Aulagnier, J. Hunink and D. Muller, "The ESPADON programme", Proceeding of RADAR' 99 International Conference, Brest, France 17 -21 May' 99.

[3] M. A. Richard et al, "The RASSP Program Origin, Concepts, and Status – An Introduction to the Issue", Rapid Prototyping of Application Specific Signal Processors, Jnl. Of VLSI SIGNAL PROCESSING SYSTEMS for Signal, Image and Video Technology, Kluwer Academic Publishers, Vol. 15, 7, Feb, 1997.

[4] G. R. Gladden,"Stop the lifecycle – I want to get off", ACM Software Engineering Notes, Vol. 7 (2), 35, 1982.

[5] B. W. Boehm,"A Spiral Model of Software Development and Enhancement", Computer, 61-72, May1988.

[6] R. Balzer, "A 15 year perspective on automatic programming", IEEE, Trans. Software Eng., Vol 11, 1257, 1985.

[7] J. Pridmore et al, "Model-Year Architectures for Rapid Prototyping", Rapid Prototyping of Application Specific Signal Processors, Jnl. Of VLSI SIGNAL PROCESSING SYSTEMS for Signal, Image and Video Technology, Kluwer Academic Publishers, Vol. 15, 83, February 1997.

[8] I. Alston and B. Madahar, "The Tool Selection Process for the ESPADON Design Environment", Proceedings of the BAE SYSTEMS Conference on Signal & Data Processing, 1-3 March, 2000, BAE SYSTEMS Research Centre, Gt. Baddow, CM2 8HN, England, Report No. YD/000140, March 2000.

[9] ESPADON Programme Deliverable, 'ESPADON: COTS Tools Assessment Summary', BAE SYSTEMS Research Centre, Gt. Baddow, CM2 8HN, England, Report No. YD/992197/D1348, Issue 2, 10th January 2000.

[10] ESPADON Programme Deliverables, 'ESPADON: Tool Evaluation Reports', BAE SYSTEMS Research Centre, Gt. Baddow, CM2 8HN, England, Report Nos. YD/992344/D1348, YD/992375/D1348, SIGNAAL 9501 114 059, Thomson 215 PV 0165, February 2000.

[11] GEDAE, Lockhheed Martin ATL, USA, http://www.gedae.lmco.com/.

[12] Ptolemy, University of California, Berkeley, USA, http://ptolemy.eecs.berkeley.edu/.

[13] Matlab, The MathWorks Inc., http://www.mathworks.com/.

[14] VSIPL, http://www.vsipl.org/.

[15] MPI-RT, http://www.mpirt.org/.

[16] ESPADON Programme Deliverable, 'ESPADON: System/Segment Design Document', Thomson-CSF DETEXIS, 1 Bld Jean Moulin, 78852 Elancourt CEDEX France, Report No. TBU.TN/ELA/ETNI, 98/1104/SSS, 28th June 1999.

[17] Alston and B. Madahar, "Rapid Prototyping", Proceedings of the BAE SYSTEMS Conference on Signal & Data Processing, 1-3 March, 2000, BAE SYSTEMS Research Centre,·Gt. Baddow, CM2 8HN, England, Report No. YD/000140, March 2000.

[18] EUROPRO Consortium, EUROPRO Final Report, Project P21040-HPCN/EUROPRO, V1.1, Thomson Marconi Sonar, Sophia Antipolis, France, May 1999.

[19] ESPADON Programme Deliverable, 'ESPADON: Validation Plan - Definition of Metrics and Reference Baselines', Signaal, Zuidelijke Havenweg 40, P.O. Box 42, 7550 GD Hengelo, Netherlands, Report No. 9501-113-932, October 1999.

[20] H. Schurer and J J. Hunink, "Rapid Prototyping of Radar Signal Processing Algorithms", IEEE, ProRISC 99 Conference Proceedings, ISBN 90-73461-18-9, 405-412, October 1999.

United States Army Commercial Off-The-Shelf (COTS) Experience The Promises and Realities

(March 2000)

James J. Barbarello
Director, Command & Control
Research, Development & Engineering Center
US Army Communications-Electronics Command
Fort Monmouth, NJ 07703, USA

Walter Kasian
Chief, Technology Planning Office
Command & Control Directorate
Research, Development & Engineering Center
US Army Communications-Electronics Command
Fort Monmouth, NJ 07703, USA

Summary: The US Army Communications-Electronics Command, commonly called CECOM, has been aggressively pursuing Commercial-Off-The-Shelf (COTS) materiel solutions for well over a decade. With that experience, CECOM has developed a strategy of "Adopt, Adapt, Develop". Through a series of case studies, this paper will explain when CECOM adopts COTS directly, adapts COTS products (by modifying as necessary to meet operational needs), and develops solutions when no COTS products will meet the Army's needs.

AMC: CECOM is one of four Major Subordinate Commands (MSCs) reporting to the Army Materiel Command. AMC as it is called, is responsible for all of the materiel used and maintained by the Army. One MSC addresses tanks and other ground vehicles. Another addresses missiles and other munitions, and Army aviation platforms. Yet another addresses all items used by the Army soldier. CECOM addresses all command & control, communications, computers, intelligence, electronic warfare, and sensor electronic systems and subsystems used in the platforms acquired by the other AMC MSCs. Based on this mission we good-naturedly say, "We don't make the platforms used by the US Army...we make them better!"

The Electronics Revolution: CECOM has been involved in things electrical or electronic for over 80 years. The last 20 years, however, has been a time of extraordinary change. The immense progress in commercial technology, especially the tremendous growth in telecommunications, computing and consumer electronics (as reflected in Moore's Law's 18-month evolutionary cycle) has changed CECOM acquisition philosophy. The products we had to spend years developing only two decades ago can now be acquired from various commercial sources.

With strong emphasis on reducing system acquisition and sustainment costs, the US military has embraced (albeit to varying degrees) COTS solutions as a way to realize those cost savings while also speeding up equipment fielding. Over those last 20 years, the use of COTS products and components in military systems and platforms has gradually increased. In ground vehicles and missiles, this use has grown slowly. In CECOM's products, the use has been surprisingly expansive.

This move towards COTS is even incorporated into the US acquisition regulations. The 1994 Federal Acquisition Streamlining act, implemented by the Federal Acquisition Regulation (FAR) in October 1995, promoted a preference for using commercial items and directed US Government procurement teams to address the acquisition of commercial items as the norm for conducting business.

Two decades ago, the US Military was a significant customer in the electronics market. As such, it could mandate to many industries (like the US microelectronics manufacturers). Today, unique military business has dwindled to just a small fraction of the overall electronics markets. In other areas, however, our buying power has increased. As a corporate entity, the Army is a major user of

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

computer systems and software. So where the Army can no longer expect microelectronics manufacturers to build devices especially for them, they can enter into arrangements with major commercial computer hardware and software suppliers (such as Microsoft Corporation) to obtain very competitive pricing arrangements.

Adopt, Adapt, Develop: CECOM strives to *adopt* commercial products and components wherever possible. This is especially desirable when the commercial product or component is offered in accordance with a commercial standard. In these instances, CECOM is not tethered to a specific manufacturer and the impact of technological obsolescence is greatly reduced.

Adoption of commercial products and components is not necessarily straightforward or risk free. For example, even when COTS is adopted, some evaluation or test is required to determine the COTS' suitability within the eventual military system. A laptop computer can be adopted for use in a command post where environmental conditions are controllable and within the scope of the product. That is not the case for extreme temperature, bounce and vibration environments. Today's commercial microcircuits are much more robust than those of 20 years ago. In most instances, even those products developed expressly for the military use such commercial devices. However, this does not apply to orbiting communications devices that would be subject to electromagnetic damage (of either natural or other nature). Failure to match the component or system to the using environment has proven costly for some commercial companies (as in the case of satellites damaged by electromagnetic effects), and could be fatal for the military.

When the COTS product cannot accommodate the using environment directly, CECOM has chosen to pursue the *adapt* route. For instance, a COTS product may have to be adapted to improve its robustness or reliability. Industry has periodically promoted products with immature technology. Easily breached security, delicate mechanical structure, or unproven software are but three of the immature characteristics encountered by CECOM in commercial products. In these instances, CECOM will work to adapt that technology to meet its customers' needs. (In the process, industry (and the commercial consumer) will benefit from applying the results of adaptation). One example is the Global Positioning System (GPS) receivers adapted by CECOM in the late 1980's from the products initially developed by Rockwell-Collins and Magellin. The resultant PLGR (Portable Lightweight GPS Receiver) made its mark in the deserts of Iraq in 1991.

Adaptation is also required when the target product must interoperate with other portions of the military host system, whether they are other commercial products or components, or items resulting from military development. As an example, Asynchronous Transfer Modem (ATM) switching became very popular a few years ago. CECOM desired to incorporate this technology into its Mobile Subscriber Equipment (MSE) communications system. The commercial ATM products, however, had to be adapted to work within the MSE system (which is a combination of commercial and military developed sub-systems).

While the *Adopt* and *Adapt* approaches work for the majority of applications, there are instances where commercial industry will not (or cannot) provide appropriate solutions. One classical example was one of our sister MSC's need for a replacement for the venerable Jeep. The Tank and Automotive Command (TACOM) attempted to adapt commercial vehicle technology with a product called the CUCV. This slightly beefed-up commercial vehicle failed miserably in field environments. TACOM then pursued development of a new vehicle that it labeled the HMMWV (the venerable "Hummer"). As in many instances, the military development satisfied the Army's need while also providing industry with a new product for their commercial market.

In fact, except in a minority of instances, military Research & Development investment does not fully fund military development. Rather, it acts as an incentive, a "seed", to entice industry to enter into a dual-use program. In such programs, the initial development funded by the military results in a future capability or product that can be commercialized. The military then uses the industry's production capacity to fulfill its needs at reduced cost, resulting in a win-win solution for both parties. A classical example is image intensifying night vision devices initially developed by CECOM and then adapted for other markets by industry around the world.

CECOM has adopted, adapted, and developed several products since it began its relentless pursuit of COTS solutions. The following four case studies explain how CECOM has used the Adopt, Adapt, Develop approach in specific instances.

Case Study #1, CHS: Since the advent of ENIAC in the 1940's, the military has striven to incorporate computing. The development of the COBOL high level programming language was also driven by the military; the model they used up until the 1980's. When the Military Computer Family of unique computers and the ADA programming language were eclipsed by the ever-expanding commercial computer industry, CECOM realized it was time for a change. In the early 1990's the Project Manager (PM) for Common Hardware and Software (CHS) found a new way to do business. Against the conventional wisdom (and significant inertia) of military developers and acquirers, PM CHS established a (then) revolutionary acquisition instrument. Called CHS-1, it was essentially an "ordering catalog" for commercial computer hardware and software. It contained products from mainstream computer suppliers and niche companies alike. With the availability of this convenient method for obtaining the latest available technology, Army PMs began to incorporate CHS products into their systems. A prime example is ABCS, the Army Battle Command Systems. In the latter half of the 1990's CECOM and its Program Executive Officer (PEO) team members proposed a new concept to the Army; digitize the battlefield. At the heart of this concept was ABCS, a system of tactical battlefield systems for maneuver control, artillery, intelligence, logistics, air defense, and fire control. CHS hardware and software (and other COTS products) were injected into the systems resulting in the highly successful Task Force XXI experiment that caused the Army to adopt the digitization strategy.

In 1999, PM CHS issued its second ordering catalog, CHS-2, administered by GTE Systems for the Project Manager. As indicated in the CHS-2 Ordering Guide, CHS-2 provides the tactical Army and the Defense Department with computer products ranging from hand held to RISC-based server class machines. CHS-2 products include life-of-contract warranty and 72-hour return/replacement, 24-hour hotline, and regional support centers located world-wide. Along with the computers, the hardware product mix includes printers, displays, storage devices and other peripherals. Hardware is supplied as Version 1, defined as commercial, Version 2, defined as rugged, and Version 3, designed for a greater degree of handling and more severe environments. Wide arrays of software products (such as operating systems, integrated business packages, programming languages and development tools) are also available. Commercial suppliers of the products include Sun Microsystems and Microsoft Corporation. As technology advances, the offered products are updated with newer ones.

1 August 1999 - Revision K

CHS-2 Ordering Guide

Quick Index	Section
Cables, V1	15.0
Cables, V2	6.0
Displays, V1	12.0
Displays, V2	3.0
Handheld Terminal Unit, V3	18.0
Introduction	1.0
Mass Storage Units and Media Drives, V1/V2	10.0
Mounting Kits, V2	8.0
Notebook Computer Unit	20.0
Power Supply, V1	13.0
Power Supply, V2	4.0
Printers, V1	14.0
Printers, V2	5.0
PSE Large Development Servers	17.0
Routers and Switches	16.0
Servers, V2	22.0
Software	19.0
Special Purpose Boards, V1/V2	7.0
Transit Cases and Miscellaneous Accessories	9.0
Versatile Computer Unit, V2	21.0
Workstations, V1	11.0
Workstations, V2	2.0

http://chs2.gte.com

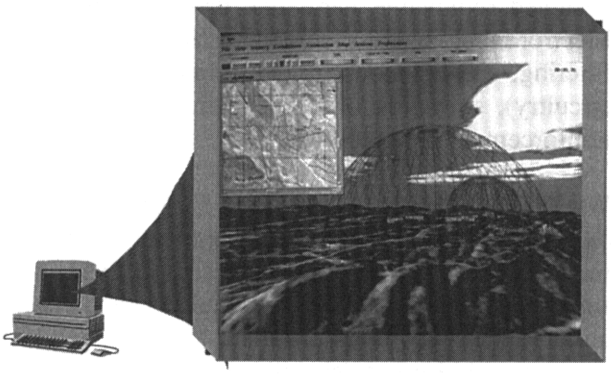
GTE

While CHS contains a mix of adopted (pure COTS) and adapted (ruggedized COTS) products, at the component level (e.g., circuit boards, memory, drives, bus, etc.), everything is adopted COTS. At the sub-system level, adaptation mechanisms include specially designed exterior cases, specially designed removable hard disk drive encasements, ElectroMagnetic Interference (EMI) gasket lens filtering, special mounting (restraints for high-risk circuitry), stiffening of printed circuit boards, and reinforcement of components.

While CHS is more or less taken for granted as the *only* way to do business in the computer arena, this was not always the case. Initially, there was significant resistance from acquirers and users alike. The acquirers warned that commercial products could not meet military operational needs and would not be sustainable. They predicted that non-military electronic components would fail miserably. They warned that relying on a vendor to repair and return products just would not work. The users worried about how they would get replacements when the products failed, and how they would continue to operate during that down time. In the final analysis, the established groups were reticent to change the way they had been doing business. They were used to risk avoidance, not risk management! Of course, it would be unfair to represent the CHS experience as all positive. There was a "break-in" or "learning" period where repair and returns were delayed, ordering wasn't as smooth as it could have been, and, in general, the full promise of CHS was not realized. But since that time, the CHS concept has been refined and has matured to an extremely effective mechanism for injecting the latest COTS technology into Army systems.

Case Study #2, Software Development. There is probably no more dynamic technology area than software applications. This area moves faster than any other and consumes more of our development and sustainment funding. While we could discuss specific applications, a more relevant area is the tools used to develop those applications (and the standards associated with them).

CECOM is responsible for bringing to the Army an ever-improving capability to visualize the battlespace and its contents. In the mid-1990's CECOM was asked to investigate the feasibility and utility of 3-Dimensional visualization. At the time, there were few, if any, commercial 3-D toolkits for software developers. Also at that time, computing power was significantly less than today and only the top-of-the-line machines (like those made by Silicon Graphics Inc. (SGI) for the Hollywood movie industry) were even close to being capable to execute real-time, 3-D visualization.



Our initial efforts used a 3-D software product (the Virtual Geographic Information System, or VGIS) developed jointly by one of our sister organizations, the Army Research Laboratory , and Georgia Institute of Technology. We adapted VGIS to meet Command & Control visualization requirements and then focused on the development of prototype applications to satisfy the user's needs. Consolidated into our Battle Planning and Visualization (BPV) system, the applications included route planning (using elevation data for slope analysis and inflection), a cross sectional 3-D view of routes, a 3-D common tactical picture, and more. At that time, the only platform capable of running these applications was the SGI series of workstations.

As time progressed, SGI released OpenGL, an industry standard, platform-independent graphics Application Programming Interface (API). VGIS, developed much earlier, used an SGI platform-specific API called IrisGL. The introduction of this new API created a conundrum. Should we continue to use IrisGL, or port all of our work to the industry standard? Since our target hardware platforms had also changed to Sun Microsystems platforms, we chose to port our applications (and VGIS) to OpenGL.

Within two years, SGI began researching graphics APIs that provided some of the advanced features we had developed in VGIS. But these features were immature, and we could not rely on them (yet). We continued to test the new SGI APIs (as they matured) while moving forward with BPV, and provided continual feedback to SGI as we did. A year later, SGI initiated a collaborative effort with Microsoft Corp. to develop a new cross-platform graphics set of API's called "Fahrenheit". Again, we participated in early trials of the new API while continuing with BPV. While SGI has scaled back its efforts on Fahrenheit, we are continuing our relationship with Microsoft by way of the Fahrenheit Beta program. We expect the final product to form the core of our future 3-D applications.

Our history and approach with the 3-D BPV system epitomizes the speed and danger associated with developing software applications. If we had waited for an industry standard toolset (instead of beginning our development with a "homebrew" toolset), we would have not been able to respond to our customers. On the other hand, if we had then closed our development environment to new tools (and not participated in Alpha and Beta testing with SGI and Microsoft), we would have encountered a "dead end". Our BPV system would have been inexorably linked to the SGI hardware platform, while our customers were using SUN and even PC platforms. BPV would not have been able to take advantage of graphic engine improvements.

Our approach was a "middle of the road" strategy. We kept an open path towards the future, but did not adopt immature products (which, in the case of SGI never matured into an actual product). As a result, our BPV is serving as the basis for new systems for our customers on various hardware platforms. And as COTS technologies become mature enough, we are continuing to integrate them.

Case Study #3, Batteries: Our first two case studies focused on software. But while software has become a major part of today's technology focus, that software needs hardware on which to execute. And that hardware needs power in order to operate. In a tactical environment, you don't have the luxury of commercial power (or even locally generated power for that matter). Thus, portable power in the form of batteries is critical to our customers.

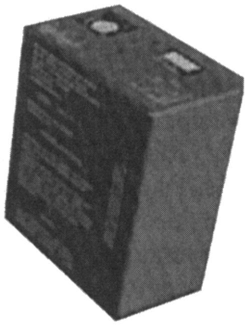
But high-energy batteries required by our soldiers' electronics gear are expensive. So much so, that in the 1990's the Chief of Staff of the Army became concerned at the high cost of batteries that the Army used on a routine basis to keep its soldiers trained and ready. He challenged AMC (and, in turn, CECOM) to reduce that cost by 50%. After some analysis, we found that the major contributor to the cost was a single Army-specific battery, the BA-5590. This battery powered the SINCGARS radio (when it was not being powered by vehicle power systems). With over 200,000 SINCGARS radios used by the Army, arriving at a solution for just the BA-5590 had the potential to meet the Chief of Staff's mandate.

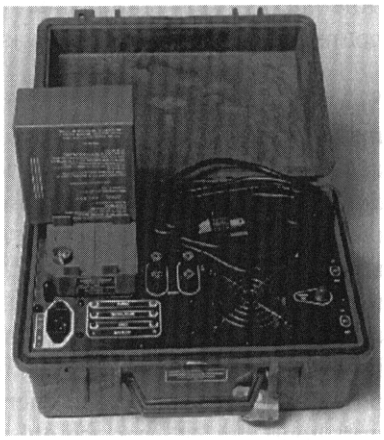
The BA-5590 was a lithium sulfur-dioxide primary (non-rechargeable) battery with high current and energy content in a relatively small (size and weight) package. Any alternative would need to maintain the same form factor and weight, and provide the same capacity so soldiers could still perform their stated missions. CECOM also faced an additional problem. Although the cells used in the batteries are essentially commercial, the battery itself is unique to the military. A commercial battery manufacturer makes more consumer "D" cells in a few days than the total yearly requirement for BA-5590's. Thus, commercial manufacturers are not interested in this "low-volume" market. Instead, the Army relies on less than five specialty houses around the world to assemble its military-unique batteries.

While the Army did improve the BA-5590 primary battery (using newer lithium chemistry), a more interesting aspect of addressing the challenge is what we did to change the "customer's" consuming habits. Over the years the Army had used both non-rechargeable and rechargeable batteries, but it rarely uses the latter for combat. And since the Army chose to train as they fight, rechargeables were not considered appropriate for training either. Years ago, this made some sense. Rechargeable batteries were not very good; they held relatively little energy, took a long time to charge, and could not reveal how much charge was left to the user.

Since that time, however, newer chemistries (nickel metal hydride and lithium ion) have become available. Nickel metal hydride technology was developed by commercial industry to replace nickel cadmium rechargeable batteries, thus addressing the new U.S. Environmental Protection Agency's regulations governing the disposal of products containing heavy metals (in this case, cadmium). Batteries with nickel metal hydride technology provide 50 percent more energy per weight (39 watt-hours per kg) than the old nickel cadmium and lead acid systems. They also suffer no "memory" problems and can be recharged at least 225 times under field conditions. Lithium ion technology was also developed by commercial industry. Their intended application was laptop computers and cell phones that demanded the most energy and power in the smallest and lightest configuration possible. Batteries made with this technology provide 100 percent more energy per weight (52 watt-hours per kg) than the old nickel cadmium and lead acid systems, suffers no "memory" problems and can also be recharged well over 225 times under field conditions

To satisfy the challenge, CECOM had to accomplish three tasks. We had to adopt commercial cells with this newer technology into our military batteries. We also had to adapt commercial charging technology to provide a field recharging system that could recharge a battery in a relatively short time (about 3 hours, versus the 12 hour charge time of the older, military developed charging systems that existed then in the Army inventory). Finally, to gain customer confidence, we had to imbed some sort of "state-of-charge" system into the new batteries so the soldier could get a relative reading of how much "life" was left in the battery. (Put yourself in the soldier's place. If you were going to literally bet your life on a battery, would you guess how much charge was left, or just throw out the one you had before you left on your mission and take a new one? If you did, you'd be throwing away a lot of unused, expensive capacity).





CECOM began by building prototype batteries with commercial (lithium ion and nickel metal hydride) cells, but in the existing military configurations.

In parallel, we contracted for the development and production of a new field charger that used commercial PIC microprocessor technology. (The PIC microprocessor is readily available, inexpensive, and has its instructions stored in erasable programmable read-only memory. This approach allowed us to change the charger's characteristics several times as we built a few chargers, took them to the field, got feedback, and made changes). Finally, we incorporated a 4-Light Emitting Diode (LED) state of charge indicator. The LEDs indicate 25%, 50%, 75%, & 100% of capacity, are inexpensive, and give just the right level of indication to the soldier.

As our experience grew, we produced more batteries and *gave* them to a series of fielded units (along with the new "rapid" chargers). Skeptical at first, the units eventually gained confidence in the new rechargeable system. They also realized that they were saving significant money by not having to buy primary batteries. After two-years of this type of trial, everyone was convinced enough for the Army to formally adopt the rechargeable system for training. In the end, CECOM and AMC more than met the Chief of Staff's challenge.

CECOM was able to adapt commercial lithium ion cell technology to its military unique batteries (similar to our adaptation of commercial microcircuits to our military computing needs). In the process, an interesting synergy evolved between CECOM and industry. While the cell technology provided greatly improved capability, it did it at temperatures only down to 0°F. Since the Army needed to operate much below that temperature, CECOM had been working on an innovative lower temperature electrolyte technology. That technology, developed by CECOM, was shared with industry that, in turn, were able to offer us further improvements in low temperature operation to -40°F.

CECOM was also able to adapt commercial charging and microcircuit technology to both the tasks of charging and determining the state of charge of military unique batteries.

But most important, CECOM was able to change consuming habits and old (albeit somewhat deserved) prejudices against rechargeable batteries. We were able to do this through a partnership with industry that provided us with not only chemistry improvements, but with the capability to work with us to develop, produce, deploy, and modify in a responsive, time-sensitive fashion.

Case Study #4, Land Warrior: In the early 1990's CECOM demonstrated a concept for bringing information technology to the soldier. This initial concept eventually became known as Land Warrior (LW). The envisioned Land Warrior system's capabilities would allow the dismounted infantryman to move and communicate rapidly on the battlefield. He would know at all times his own location, those of his squad members and of the enemy, regardless of terrain or weather conditions; as well as what his squad or team leader expects him to do. Land Warrior would represent advancement in effectiveness over the way today's infantry rifle squads perform collective tasks, since today they still rely heavily on verbal communications (shouting at each other) and hand and arm signals to perform collective tasks.

To achieve this capability, a contract was awarded in the 1990's to Hughes Aircraft Corp. (later acquired by Raytheon Company) to mature the concept into a fieldable system. Based on various requirements from the user (and the fact that the commercial sector was in its infancy in the wearable computer market, and laptop/notebook PCs were about the size of briefcases), Hughes set out to apply a commercially available microprocessor chip, but develop a unique, real-time operating system. After several years and over US $100 Million, the program had not progressed to the fieldable system stage and the US Congress was considering terminating Land Warrior.

In late-1998, a new PM was assigned (COL Bruce Jette, PM Soldier). Although the PM was associated with another MSC, he came to CECOM (where he had previously served) and asked us to perform a third party assessment of the LW program. Our assessment showed a high risk with the existing approach. The PM then contracted with a Silicon Valley firm to perform another independent assessment (this time with a purely commercial eye and focusing on the technology being employed). That firm came to the same conclusion as CECOM. In 1999, the PM asked that Silicon Valley firm to quickly put together a demonstration of what might be possible with today's technology. He also asked CECOM to put together a support cell to bring our technology expertise and COTS-based thinking to LW.

The resultant system is a combination of the adopt, adapt, and develop aspects. Before we identify which is which, let's take a look at the new system itself.

At the core of the integrated Land Warrior system is a small, wearable, computer-radio subsystem, mounted on the soldier's lower back. The current version of Land Warrior uses a small, portable commercial-based IBM-compatible computer, and a Windows-based operating system. This shift to an open commercial architecture will significantly reduce the cost and effort to continually develop and sustain the software. It will also make future product upgrades easier. Finally, it will help us to fine tune and tailor the system, both as technology advances, and as users adapt to the system, identify new needs and propose new capabilities.

The computer displays imagery that the soldier views through a helmet-mounted, monocular viewfinder covering one eye. The Land Warrior soldier sees a miniature computer screen – a "heads-up display", that shows digital maps, graphics, and text in a Microsoft Windows, pull-down-menu format, as well as imagery from the Thermal Weapon Sight or daylight video sight. The view he gets is from the direction at which he points his weapon. The display allows the soldier to find a target and shoot his weapon accurately from a concealed position using either the thermal or the video sight, exposing only his hands. He can even fire his weapon from behind the corner of a building without exposing his head.

The "mouse" control for the computer's menu-driven displays is a small button on the side of the weapon that the soldier manipulates using the fingers on his trigger hand. Each soldier, using a helmet-mounted microphone that sits in front of his mouth, can talk with others in his squad via secure voice radio, akin to an intercom system on an aircraft. Using the pull-down menus, he can digitally transmit spot reports of enemy activity or capture and send a video or thermal image of a target, either to squad members or to higher echelons, all using his mouse control. The soldier can even digitally transmit an automatically formatted "call for fire" (for example, to the artillery), and relay the target's coordinates at the touch of his fingers. In contrast, today's infantrymen must use paper maps and verbally convey spot reports, which are ultimately relayed by radio up the chain of command by the squad leader, and through echelons, before a digital linkage can be established.

The Land Warrior squad leader and his two fire team leaders can communicate with squad members from covered positions using voice radio, or silently using text messages. They are also equipped with a hand-held, flat-panel display that can be used to send orders silently. For example, the squad and team leaders can "write" on the hand-held map display to overlay graphics or short text, such as circling the target objective and marking the route to it. These graphics can then be transmitted to squad members' heads-up displays.

A built-in Global Positioning System receiver provides the soldier's position location to the computer, which also receives location reports from other soldiers in the squad, and are shown as icons on a digital map display. The Land Warrior can use the laser range finder to pinpoint a new enemy position, which then appears as an icon on all of the squad's map displays.

The computer is connected to the Thermal Weapon Sight, which is atop his standard rifle. The computer is also linked to a combined laser range finder and digital compass, with a video TV camera sight (also mounted on the rifle).

To see if we were on the right track with the users, 13 systems were built and delivered to soldiers for their evaluation. Although the initial systems did not meet all of the user's requirements, they did meet many of them. Today, an iterative process is in place to continually evolve and build successive and successfully functioning Land Warrior systems, with a Windows-based, IBM PC-compatible COTS-adapted computer, with commercial interface standards, packaged within a rugged case. In fact, 55 experimental systems will be demonstrated during a Joint Contingency Force exercise, in September 2000. This approach of getting products into the hands of soldiers quickly so they can provide feedback in real time to tell us what's right with the system and what needs to be better, is key to applying information age technology.

So now, let's take a look to see what was adopted, adapted, and developed. In the adopt arena, the computer was replaced with COTS computer components (albeit reconfigured in a customized case). In the process, the system gained processing speed, storage (from 500 MB to over 1.5 GB), and the ability to interface with today's peripherals over IEEE standard interfaces. The software is now being developed with commercial software development tools and has the look and feel of a "windows" environment that many young soldiers are intimately familiar with these days. That software will execute in a COTS windows-based environment.

In the adapt arena, the GPS location device is a COTS-adapted product that will include the greater precision of military GPS with protection from hostile intent. Also being adapted is a COTS heads-up display that replaces the older plasma technology but will be environmentally hardened to withstand the rigors of the foot soldier (including the ability to survive when the soldier parachutes into the area of engagement). Further adaptation is occurring in the Local Area Network (LAN) arena. Soldiers in a squad are connected via a COTS-based wireless LAN that will have higher levels of security than commercially available.

In the develop arena, the soldier's weapon is a standard Army issue product that will eventually be replaced by a new generation weapon currently under development. Then there's the laser rangefinder (developed by Raytheon) and a standard Army-developed and Army inventory thermal weapon sight.

This combined strategy resulted from the needs to inject the latest technology into the system, be interoperable with the ABCS system, and accept the realities of what can be accomplished *now*. A not-insignificant aspect of how this was accomplished in such a relatively short time (when lengthy, previous attempts were not successful) deals with working with the user. Although operationally desirable, many of the requirements cited by the user were driving costly, non-COTS solutions. By working with the user to perform a no-nonsense needs/benefits tradeoff, the PM and CECOM were able to redefine the system. For instance, the user had identified that the time between a soldier being identified and being informed of that fact was originally less than 0.2 seconds. That single requirement drove Hughes to opt for a real time operating system, since the notification had to be routed through the computer. After pragmatic consideration that the soldier's physical response time was significantly greater than 0.2 seconds, the user agreed to a longer response time. This allowed the PM to adopt a commercial computer with a commercial operating system.

Clearly, Land Warrior represents a microcosm of thoughtful application of the adopt, adapt, develop strategy. But the key lesson learned is that human communication, and not technology, is the critical factor in how (or even if) a program is able to reap the advantages of COTS.

Lessons Learned: While the scope of intensity will vary with the technologies being addressed, the pursuit of COTS is clearly preeminent in CECOM's lexicon. But blind adoption of COTS is neither technically desirable nor fiscally sound. Rather, we pursue balance between adopting, adapting, and developing when the other two options do not meet our needs.

Adopting is not free. Funding must be reserved for testing the to-be adopted COTS. It must be capable of fitting within the current system constraints (and every existing system will have constraints). With these conditions met, adopting is the quickest and least expensive approach. A non-fiscal benefit is increased customer satisfaction, since customers will continually compare your solutions to what they can acquire on the open market.

Adapting in a military environment is pragmatically the most common solution. Adapting will usually include some associated level of adoption at either the component or sub-system level. Adapting is an optimum mix of leveraging commercial investment and the customer-environment understanding of your organic workforce. While adapting may not seem as quick or inexpensive as adopting, when the user environment is factored in, it is.

Developing must be reserved for those unique circumstances where no commercial solution can form either the total answer or a foundation for the answer. Development can no longer be considered a stand-alone effort; in today's fiscal environment, this is a sure recipe for disaster. Rather, development must be pursued as a partnership where industry is "seeded" with an initial investment (of money or technical knowledge). Properly nurtured, that seed will grow into a solution that services your customers and provides a cost-effective manufacturing base.

Only One Piece Of The Puzzle: Technology, that is. Through its significant experience in the pursuit of COTS, CECOM has learned that customer requirements and expectations are as important (or possibly more important) than the pure technology.

As with CHS (Case Study #1) and batteries (Case Study #3), customer pardigms must be understood and thoughtfully modified. Nothing breeds success like success. Early, moderate successes are much more important than the 100% solution that takes too long.

Many military customers identify requirements without the benefits that moderate trade-offs could bring. As with Land Warrior (Case Study #4), a simple trade-off in response time can allow system design that opens up the system architecture, provides a better user interface and, in general, holds the potential for greater, longer term user satisfaction.

The Difference Is Blurring: As we indicated in Case Study #2, the rapid and fluid software environment that many associate exclusively with the commercial sector is just as applicable to the military. In fact, customer expectations demand no less. And this shift extends to the information technology hardware associated both directly with software (like computers and peripherals) and indirectly (like software-programmable radios). The decisions we make in development drive the long term future of the resultant product. Selecting a COTS solution is not a trivial matter and can drive life cycle costs significantly. To make the most informed decisions, we must monitor COTS product forecasts like the stock market, and be ready to shift when necessary, or potentially pay the price for remaining static. A reminder of this was a past decision to equip all of the Navy's recreation centers with higher quality "Betamax" VCRs, instead of VHS, right about the time VHS became the consumer product of choice. Inclusion of commercial products can potentially reduce life cycle costs in military system or platform development, by leveraging in the commercial product's economies of scale, but only if there is an "active" economy of scale to work with.

Conclusion: The reality is that the significant investments being made by the commercial sector in Information Technology are orders of magnitude greater than the US military can afford to drive or influence. CECOM has recognized this and embraced an adopt, adapt, develop philosophy. We leverage commercial investment by anticipating (through technology forecasts) and building meaningful, regular interactions with industry. In these ways, we can better anticipate the direction the market is going so we can match technology trends to soldier's needs.

But as the four case studies presented infer, any decision "today" to adopt, adapt or develop, may be different "tomorrow". There is no specific formula we can calculate because the variables are continually changing. But some things *are* constant. We ensure continual interaction with the customer. We value continual technical curiosity and acumen. We foster a continuing demand to not stick with yesterday's process. And, above all, we continually keep a balance between being the earliest-adopter and one who stands still. Because in the business of equipping the US soldier with the best technology in the world, the consequences of doing it wrong (or doing it too late) can be, literally, deadly.

The Coordinated Defence Role in Civil (Telecom) Standardisation

(February 2000)

JP Thorlby
NATO C3 Agency
PO Box 174
2501 CD The Hague
The Netherlands

Abstract

The "ruthless pursuit of COTS" is increasing the penetration of unmodified COTS[1] technology and standards in the military domain. Therefore, as the defence community becomes more reliant on off the shelf products and standards, it is increasingly a stake-holder in the results of the civil process. This should lead to a motivation to be a proactive participant in the civil process by which the civil standards (and technology) are developed.

This paper presents the outcome of a recently held workshop (29th November 1999) organised by the NATO C3 Agency and hosted by the European Telecommunication Standards Institute (ETSI). The agenda, report and presentations are available at http://www.nc3a.nato.int. This paper discuses ETSI specifically, but the arguments and principles also apply to other standards fora.

It was proposed that there should be a coordinated action within the defence community of the Alliance with respect to civil standards which will encourage the emergence of a harmonised defence market for civil telecommunication products (COTS).

This paper will discuss the possibilities and significance of defence requirements capture within the context of civil telecommunication standards development.

Acknowledgement

Whilst the content of this paper is the responsibility of its author, the whole is necessarily influenced by the many views excellently articulated by the workshop speakers and participants.

Introduction

The role and benefits of standards (in the context of this paper, telecommunication standards specifically) are generally accepted in both the civil and defence communities. In the military context, standards enable interoperability between systems (in particular, systems of different nations operating in a coalition setting), reduce dependence on single suppliers and permit systems to be upgraded whilst enabling interoperability with legacy systems.

In the civil market, standards enable interoperability between competing vendors, increasing the effective size of a market which provides economies of scale, in turn enabling reductions in price which in turn fuels market growth. One spectacular recent example is the mobile technology GSM, which through a regional standardisation activity created an initial market across Europe which is now global. The market size has turned the user (mobile) terminal into a commodity product (terminals are often given away). The market has grown from its inception (1991) to 450 million users currently, and is projected to grow to 1.6 billion users by the year 2010. The total number of mobile (all technologies) users is projected to exceed the number of fixed telephones by the year 2004.

Defence users already make significant use of civil standards and COTS equipment. This trend is likely to increase in the future due to continuing downward pressure on defence spending, increased 'operations other than war' (e.g. 'Peace Support Operations') and rapid advances in technology. One example within the NATO context is the adoption of ISDN standards in the NATO core network (NCN).

If these trends are accepted, defence users (and operators) should recognise that they are 'stakeholders' in civil standards and it therefore seems logical that they take a greater interest in the process by which those standards are derived, and take steps to enter their specific requirements alongside all the other user requirements.

This paper first describes a workshop organised to address this subject area, drawing from the presentations and associated discussion. The paper then goes on discuss the issues which arise, finally making some conclusions and specific proposals for further activity.

[1] Commercial Off the Shelf Technology

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

Workshop

The NATO C3 Agency organised a workshop entitled "Defence Markets for Telecom Standards and Technologies". The workshop addressed the following general questions:

- How can the ETSI community capture the requirements of the military community in an effective manner?
- How can the NATO community understand the working methods and procedures of ETSI?
- How can industry effectively relate defence market requirements to civil market requirements?
- How can national defence agencies be facilitated to work within civil standards bodies?

The workshop brought together a representative community in an environment which encouraged free discussion It was structured around a framework of presentations from both civil and defence communities.

There were 55 delegates registered, from 12 countries and 32 organisations. The breakdown by type of the organisations is shown in figure 1.

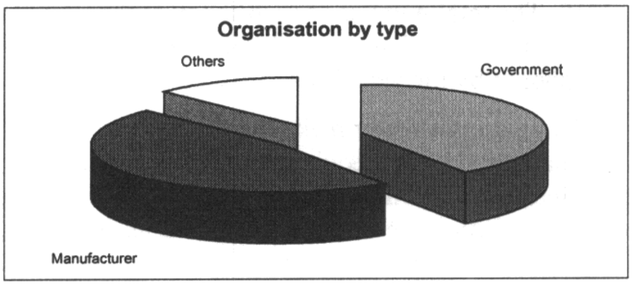


Figure 1: Organisation by Type

The breakdown of representation by country (of organisation rather than individual) is shown in the figure 2.

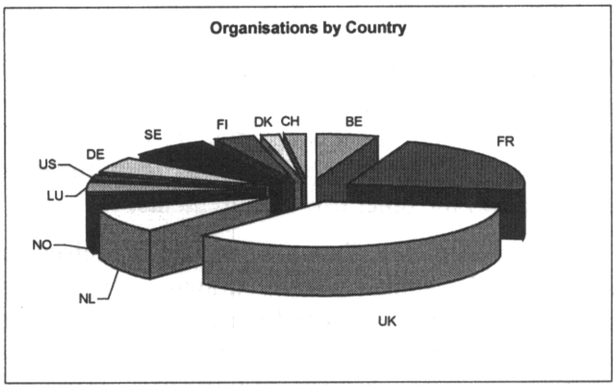


Figure 2: Organisation by Country

ETSI

ETSI is a regional telecommunication standards organisation. Its work programme is driven (and mostly funded) by its membership (730 member organisations from 50 countries, of whom 51% are manufacturers) and they in turn are driven by the perceived 'market requirement'.

Considering the diversity of members, many of whom will eventually compete in the market place to supply products and services, the challenge is always to reach consensus - which ETSI defines as "the lack of sustained opposition". Occasionally voting is required on technical issues when there is a failure to reach consensus.

ETSI is open (its work and standards are freely available at http://www.etsi.org) with only current temporary working documents restricted to member organisations. ETSI works in partnership and has cooperation agreements with many other organisations, for example; UMTS Forum, GSM Association, ATM Forum, IETF, WAP, IPv6.

The desire by several regions to work together for a single global standard for mobile communications, building on the success of GSM (an ETSI standard) took ETSI beyond its normal geographical region and so the 3rd Generation Partnership Project (3GPP) was established. This was created outside ETSI and overcame the limitation that ETSI members without a European base have no right to vote (although Associate members may participate in the technical work).

The 3GPP (http://www.3gpp.org) is a partnership of regional standards developing organisations; ETSI (Europe), T1 (US), ARIB (Japan), CWTS (Canada), TTA (Korea) and TTC (Japan) along with market representation partners (GSM Association and UMTS Forum). This shows the range and flexibility of cooperation possible and already practiced.

Standardisation therefore offers an open consensus building forum for pre-competitive R&D and thus also provides a 'load sharing' mechanism for the development of technology in addition to the actual writing of the standards.

ETSI is currently undertaking a review of its future role, and giving careful consideration to how it can best respond to the 'internet challenge', and work with global partners and streamline (speed up) its standardisation process. One recommendation which is currently for further study is the possibility of ETSI facilitating the creation of closed special interest groups, which create their own rules.

ETSI is interested to improve working with the defence community.

Requirements Capture in ETSI

ETSI operates a contribution driven culture. Technical meetings are open to full and associate members and others by arrangement. Where these arrangements prove inadequate for the target market, a partnership (such as the 3GPP) may be established external to ETSI.

Standard development in ETSI undergoes a 'requirement capture' phase' and often there is a sub-committee (or working group) dedicated to this process. For example in the Technical Committee (TC) SMG (Special Mobile Group), sub-technical committee 1 (SMG1) is responsible for specifying service requirements, which are then developed by the remaining sub-technical committees. In ETSI Project (EP) TETRA, working group (WG) 1 is responsible for requirements capture, and the requirements are then translated into standards by the remaining working groups within EP TETRA.

The emerging ETSI DIIS (Digital Interchange of Information and Signaling) standard is targeted at small communities of users (taxi firms, private security guards, retail outlets, etc.) who collectively do not have the resources (or expertise) to attend standards meetings. In this case the manufacturers need to do extensive market research to capture the user requirements (note that market research is closely related to requirements capture). The manufacturers are using an 'integrated product research process' in the standards forum.

The opportunities to enter requirements into ETSI standards are therefore:

- Direct participation in the technical meetings.
- Representations to participating manufacturers.

The effectiveness of the contributions are based on:

- The associated business case (cost of implementation versus increased market size).
- Technical merit.
- Being present and making the case.

The military have a broad range of requirements, many of these map readily onto existing civil market requirements (sometimes they may differ only in the terminology used). Some defence requirements may only have a minor (if any) cost impact, or actually improve the civil market opportunities. For Example, the workshop discussed the civil requirement for priority and preemption: the military view that there was not a civil requirement was countered by the civil view that there was, and that some ETSI standards already incorporate these features (e.g. TETRA and GSM-R).

Technical committees in standards fora often have difficulty receiving direct input from users, and user representatives may therefore make a valuable and welcome contribution.

Advantages

The advantages for the defence community of working closely with civil standards are manifold. Standards are already seen and accepted as an important component to achieve interoperability. They facilitate interworking and interoperability between current, future and legacy systems (standards tend to evolve more slowly than technology). They facilitate interoperability between different vendors equipment, increasing the freedom for competitive initial procurement and competitive mid-life upgrade. They facilitate interoperability between different nations in coalition operations. Standards provide economy of scale (through cooperation) in the development of new technology. Standards give consumers increased confidence to invest in new technology, thereby encouraging market development. Standards are the essence of communication systems (in that communication systems must interoperate in order to communicate!)

Standards fora offer a ready made environment for consensus building. They provide the involvement of industry for free (and access to key people within companies who might otherwise be difficult to get hold of). They provide a place to expose, discuss and trade off and refine requirements, in cooperation with and with inputs from industry - in the spirit of recent approaches to 'smart procurement'.

COTS equipment which has been developed in conformance to standards which have successfully captured or (more realistically which have partially captured) defence requirements will more likely be a cost effective solution than if those standards are developed without any input from the defence community.

Being present in and aware of developments in standards fora additionally gives advance knowledge of new developments, typically 1-2 years in front of the appearance in the market of products. It also gives insight into the technical capability, technical options and technical limitations of the developing technology.

Standards are only a part of technology development and may offer many implementation options. Defence use of equipment built to a particular standard may be facilitated by simply specifying a recommended option profile (i.e. which set of options best serve the defence application). The ability to do this requires familiarity with the standards and the likely stance by the various manufacturers on each option. Good judgement will be facilitated by being present when the standards requirements were discussed and the options presented for inclusion.

Many organisations participate in standards for credibility - to be present when the standards are formulated - 'table stakes'.

NATO and Standards

NATO has for many years recognised the importance of standards, and the necessity to use existing standards wherever possible and it works towards this end within the NATO C3 Organisation (through the various sub-committees) through the production of 'standards agreements' - STANAGs.

In areas where existing standards are not sufficient the nations may work together to create new standards, within the NC3O. The alternative is to accept proprietary technology.

Whereas NATO policy with respect to existing standards is clearly defined, the action with respect to standards which only partly meet requirements is not clearly defined and the methods of dealing with the associated 'requirements gap'.

Challenges

Drawing together for mutual advantage the standards creating activities of NATO and standards developing organisations (for example ETSI) presents several challenges.

The language, culture and processes of NATO standardisation may appear quite different from the language, culture and processes of civil standards fora, such as ETSI. The technical experts and operational requirements community in NATO may not be familiar with the working procedures and methods of ETSI.

The members of civil standards bodies are not in general familiar with the defence market requirements, decision making or procurement processes. Even where a manufacturing company has interests in both civil and military markets, often there is a dividing line between these two communities within the company.

The first challenge is to build a bridge between these two communities. To relate the commercial 'business process' to the military business process ('doctrine'), the commercial 'market research' and 'market requirements' terminology to the military 'operational requirements' and 'user requirements'. The challenge is to overcome the 'natural resistance' to commercial technology which exists in the military community, to separate out those technical areas where differences of approach are not reconcilable and to work together in all the other areas. The military always see an enemy, and therefore will never willingly expose a weakness. This concern must be addressed for the defence community to work effectively in civil standards fora. The challenge is to separate the parts of a defence requirement which have security sensitivity, from those which do not.

The challenge which exists in both civil and military communities is the reluctance to own (provide resources to support) the participation (effort) in activity related to standards, since this participation makes no *immediate* contribution to profit (or operational capability) and consumes resources. The value of the activity may only be realised in the long term, and is difficult to quantify in general. This is exacerbated when the outcome of a democratic consensus building process (which standards development inevitably is) may not meet 100% of the requirements of one organisation.

It is important to recognize that because defence markets are relatively small in numbers terms, the ability to guarantee success in having requirements adopted does not exist. This often gets translated into an objection to any involvement in the civil standards process, therefore denying all of the benefits because 100% success cannot be guaranteed. The challenge is to determine a balanced approach - measuring resources against realistic outcome. Standards are created for many user groups which may be individually small (for example the DIIS standards described above). Many companies participate which are small. No single organisation (user, or manufacturer) can reasonably expect to achieve 100% of its objectives in a consensus building forum, but this does not prevent them participating.

The challenge is to maximise the effective coordination within the defence community, in order to increase the chance of success. This implies agreement within the defence community on at least some requirements, followed by coordinated working within the chosen standards body.

Potential Areas for Collaboration

This section identifies possible areas for future closer working between civil standards bodies and the defence community within NATO. Some areas are clear opportunities to develop or 'fine tune' existing standards, some areas are more closely related to research activity and the scientific programme of work.

Strategic Tactical Interoperability

The development of the standards agreement for the Digital Strategic Tactical Gateway (DTSG) which is essentially an interface specification between ISDN and tactical networks (which may include STANAG5040 or STANAG 4206 gateways) may result in requirements for small enhancements to the ISDN standards. The DSTG (STANAG 4578) could be developed inside the framework provided by (for example) ETSI, alongside 'change requests' to ISDN specifications.

Advanced Network Architecture

ETSI has a number of projects which are addressing and developing advanced network architectures as part of new standardisation activities. For example the 3rd Generation Partnership Project (3GPP) is necessarily addressing the evolution of the core network technology (currently ISDN based), taking into account (for example) internet (IP), mobile internet (MobileIP), and ATM. ETSI Project TIPHON focuses on voice communication and related multimedia aspects as required to enable interoperability within IP based networks and with other types of networks. ETSI Project BRAN deals with broadband radio access networks.

Personal Communication Services (PCS)

ETSI is responsible for the GSM standard, is a partner in 3rd Generation standards (3GPP) development, is responsible for TETRA (which is being considered by many nations for various military applications.) Aspects of TETRA implementation may still be open for standardisation work, particularly in the area of security. The proposed broadband successor to TETRA is in the very early stages of development (DAWS) and therefore now open to users requirements input.

Summary

Telecommunications is developing at a fast pace, particularly in the civil domain.

Defence users increasingly make use of civil standards and equipment procured to civil standards and therefore are 'stakeholders' in those standards.

The resources available to the defence community to invest in proprietary solutions in order to fill the 'requirements gap' are diminishing.

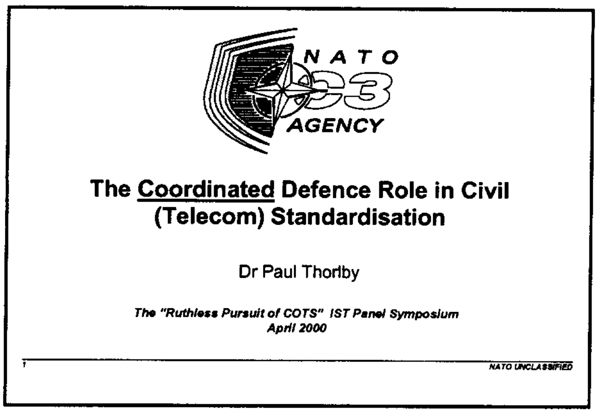
Coordination within the defence community with respect to working with standards will result in efficiency gains and increased effectiveness.

Standards organisations provide a consensus building forum with industry.

Conclusions

NATO and the NATO nations should give due consideration to areas of technical work which may be effectively progressed in cooperation with, or within civil standards bodies such as ETSI, and reduce to a minimum the 'proprietary standardisation' activity which currently occurs.

A coordination activity should be started within NATO to facilitate and encourage defence participation in civil standard bodies.



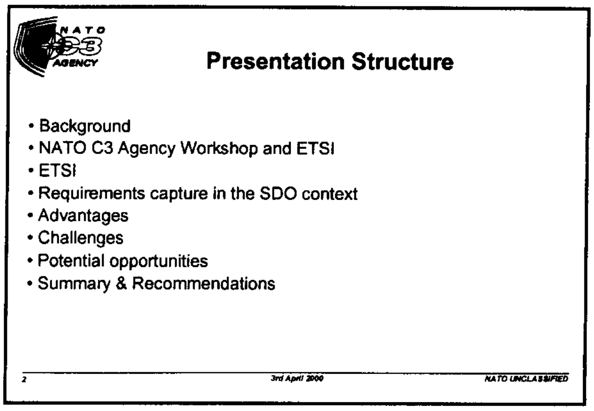
NATO C3 AGENCY

The Coordinated Defence Role in Civil (Telecom) Standardisation

Dr Paul Thorlby

The "Ruthless Pursuit of COTS" IST Panel Symposium
April 2000

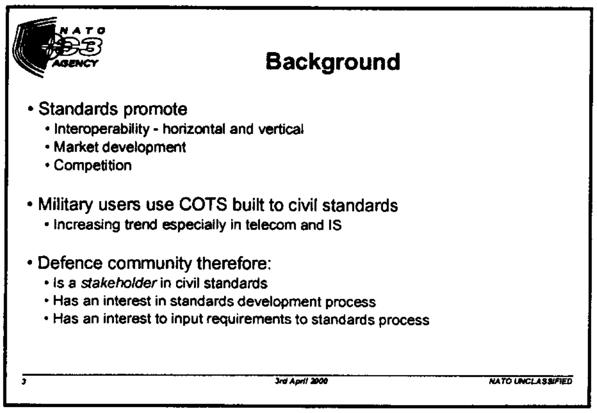
1 NATO UNCLASSIFIED



Presentation Structure

- Background
- NATO C3 Agency Workshop and ETSI
- ETSI
- Requirements capture in the SDO context
- Advantages
- Challenges
- Potential opportunities
- Summary & Recommendations

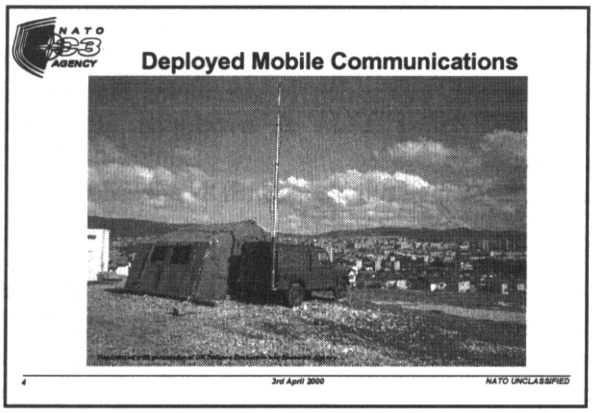
2 3rd April 2000 NATO UNCLASSIFIED



Background

- Standards promote
  - Interoperability - horizontal and vertical
  - Market development
  - Competition
- Military users use COTS built to civil standards
  - Increasing trend especially in telecom and IS
- Defence community therefore:
  - Is a *stakeholder* in civil standards
  - Has an interest in standards development process
  - Has an interest to input requirements to standards process

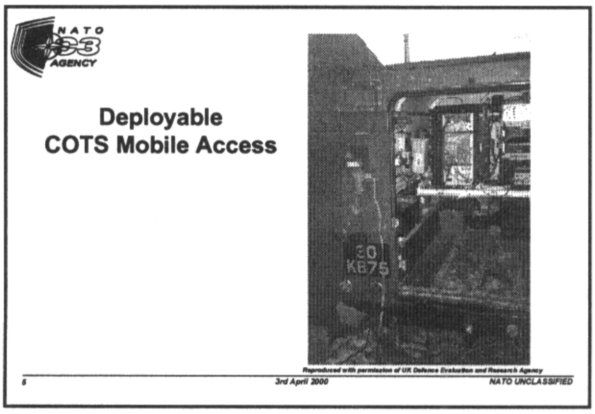
3 3rd April 2000 NATO UNCLASSIFIED



Deployed Mobile Communications

Reproduced with permission of UK Defence Evaluation and Research Agency

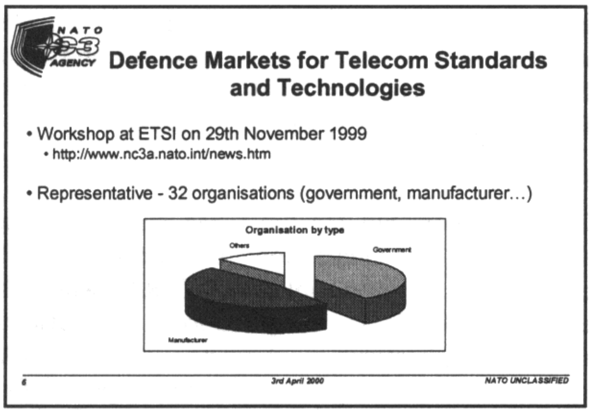
4 3rd April 2000 NATO UNCLASSIFIED



Deployable COTS Mobile Access

Reproduced with permission of UK Defence Evaluation and Research Agency

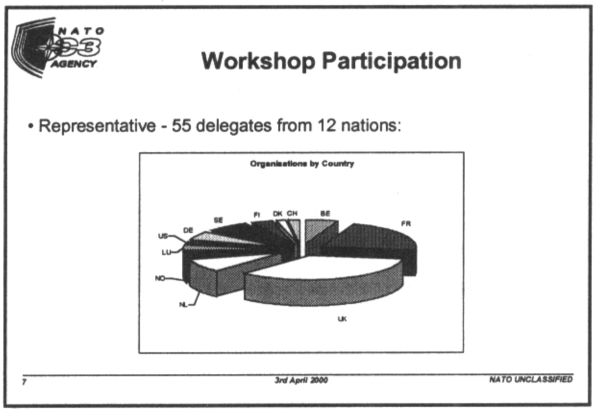
5 3rd April 2000 NATO UNCLASSIFIED



Defence Markets for Telecom Standards and Technologies

- Workshop at ETSI on 29th November 1999
  - http://www.nc3a.nato.int/news.htm
- Representative - 32 organisations (government, manufacturer…)

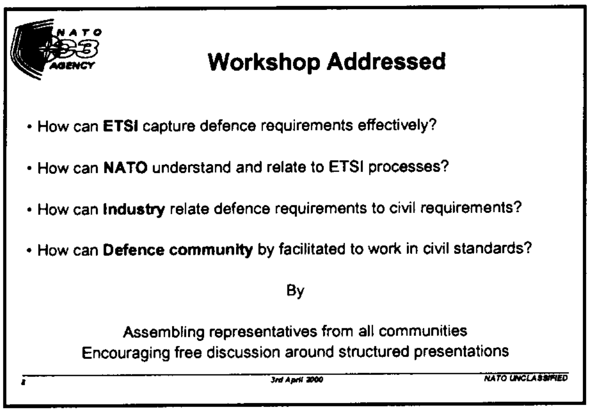
6 3rd April 2000 NATO UNCLASSIFIED



Workshop Participation

- Representative - 55 delegates from 12 nations:

7 3rd April 2000 NATO UNCLASSIFIED



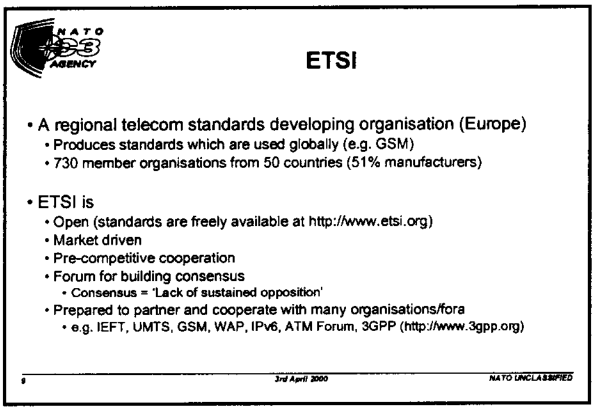
Workshop Addressed

- How can **ETSI** capture defence requirements effectively?
- How can **NATO** understand and relate to ETSI processes?
- How can **Industry** relate defence requirements to civil requirements?
- How can **Defence community** by facilitated to work in civil standards?

By

Assembling representatives from all communities
Encouraging free discussion around structured presentations

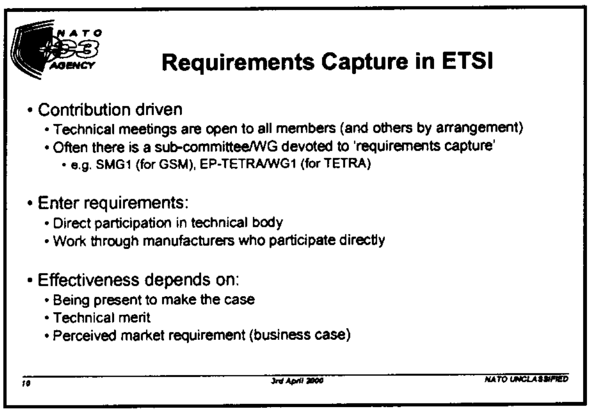
8 3rd April 2000 NATO UNCLASSIFIED



ETSI

- A regional telecom standards developing organisation (Europe)
  - Produces standards which are used globally (e.g. GSM)
  - 730 member organisations from 50 countries (51% manufacturers)
- ETSI is
  - Open (standards are freely available at http://www.etsi.org)
  - Market driven
  - Pre-competitive cooperation
  - Forum for building consensus
    - Consensus = 'Lack of sustained opposition'
  - Prepared to partner and cooperate with many organisations/fora
    - e.g. IEFT, UMTS, GSM, WAP, IPv6, ATM Forum, 3GPP (http://www.3gpp.org)

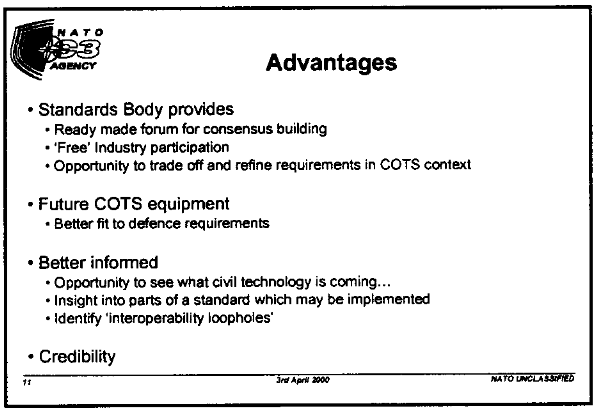
9 3rd April 2000 NATO UNCLASSIFIED



Requirements Capture in ETSI

- Contribution driven
  - Technical meetings are open to all members (and others by arrangement)
  - Often there is a sub-committee/WG devoted to 'requirements capture'
    - e.g. SMG1 (for GSM), EP-TETRA/WG1 (for TETRA)
- Enter requirements:
  - Direct participation in technical body
  - Work through manufacturers who participate directly
- Effectiveness depends on:
  - Being present to make the case
  - Technical merit
  - Perceived market requirement (business case)

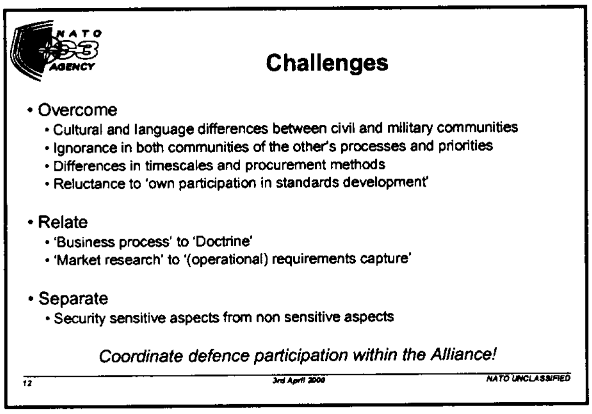
10 3rd April 2000 NATO UNCLASSIFIED



Advantages

- Standards Body provides
  - Ready made forum for consensus building
  - 'Free' Industry participation
  - Opportunity to trade off and refine requirements in COTS context
- Future COTS equipment
  - Better fit to defence requirements
- Better informed
  - Opportunity to see what civil technology is coming...
  - Insight into parts of a standard which may be implemented
  - Identify 'interoperability loopholes'
- Credibility

11 3rd April 2000 NATO UNCLASSIFIED

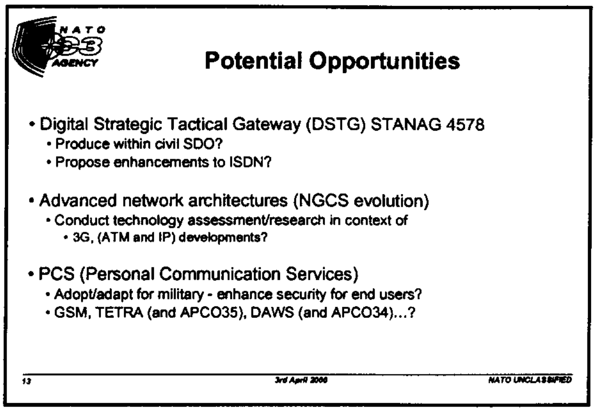


Challenges

- Overcome
  - Cultural and language differences between civil and military communities
  - Ignorance in both communities of the other's processes and priorities
  - Differences in timescales and procurement methods
  - Reluctance to 'own participation in standards development'
- Relate
  - 'Business process' to 'Doctrine'
  - 'Market research' to '(operational) requirements capture'
- Separate
  - Security sensitive aspects from non sensitive aspects

Coordinate defence participation within the Alliance!

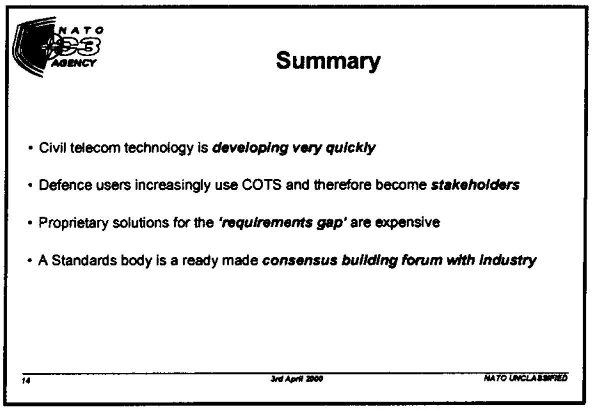
12 3rd April 2000 NATO UNCLASSIFIED



Potential Opportunities

- Digital Strategic Tactical Gateway (DSTG) STANAG 4578
  - Produce within civil SDO?
  - Propose enhancements to ISDN?
- Advanced network architectures (NGCS evolution)
  - Conduct technology assessment/research in context of
    - 3G, (ATM and IP) developments?
- PCS (Personal Communication Services)
  - Adopt/adapt for military - enhance security for end users?
  - GSM, TETRA (and APCO35), DAWS (and APCO34)...?

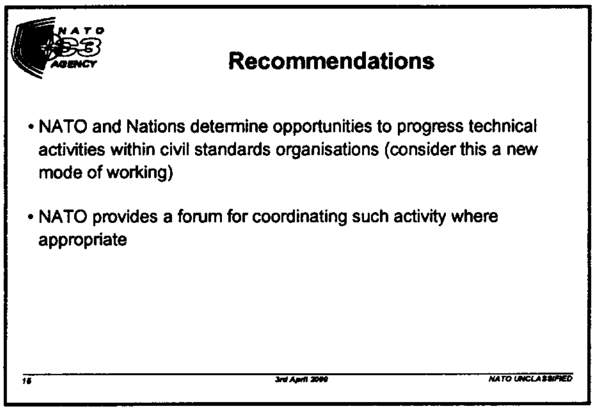
13 3rd April 2000 NATO UNCLASSIFIED



Summary

- Civil telecom technology is ***developing very quickly***
- Defence users increasingly use COTS and therefore become ***stakeholders***
- Proprietary solutions for the ***'requirements gap'*** are expensive
- A Standards body is a ready made ***consensus building forum with industry***

14 3rd April 2000 NATO UNCLASSIFIED



Recommendations

- NATO and Nations determine opportunities to progress technical activities within civil standards organisations (consider this a new mode of working)
- NATO provides a forum for coordinating such activity where appropriate

15 3rd April 2000 NATO UNCLASSIFIED

Risks by Using COTS Products and Commercial ICT Services

(March 2000)

Susanne Jantsch
IABG mbH
Einsteinstrasse 20
D - 85521 Ottobrunn, Germany

Introduction

Among the requirements influencing today's procurement of new information and communications systems, the most prominent are

- cost effectiveness
- use of the latest developments in information and communications technology (ICT)

through the whole lifetime of a system. This can no longer be achieved in procurement procedures as they used to be, with long planning and development phases, resulting in proprietary products based more and more often on out-dated technology at the time they go operational. Also, storage or provision of spare parts for and maintenance of such fully or mainly proprietary systems, as well as the education and training of personnel for their operation and maintenance, are increasingly cost intensive.

The alternative and inevitable approach is the consequent use of COTS products, allowing for easy and timely release changes and introduction of new hard and software versions when they come to market, paired with the consequent outsourcing of all those services which are available with comparable or higher quality by non-military providers, allowing usually to choose among competitive offers.

However, though on first view this new way of procurement seems to perfectly meet the above mentioned requirements for cost effectiveness and application of the latest ICT developments, there is also a new class of risks to be identified and dealt with.

After summarizing the eminent advantages of the consequent use of COTS products and outsourcing, this paper will address the risks that have to be considered and finally point out methods to improve confidence in how to use "unsecure" products and services.

Benefits of COTS Products and Outsourcing

Innovation rates in modern ICT keep decreasing at a breathtaking pace, while at the same time new developments continuously broaden the spectrum of service details and technical features waiting to be introduced into new or refined products. Integration and diversification occur in parallel, allowing to design and produce in large numbers products adapted or adaptable to very specific customer requirements.

As an example, we see today mobile phones more and more equipped with services / interfaces for services like WAP and SMS, allowing to use a piece of hardware originally designed to communicate via speech to send and receive written messages and to retrieve information from the internet. On the other hand, the spectrum of available mobile handsets and contracts differing in service details leads to such a fast change of products (as a combination of hardware, software, and service) offered by service providers to the enormously increasing number of mobile phone users all over the world that a market analysis may easily be outdated within three months.

This example illustrates the ever changing variety of often highly competitive products openly available on the ICT market.

Competition helps in both keeping the prices low or bringing them down and in the products constantly being made more attractive by add-ons, by featuring the latest technological developments, and in the case of complex systems by add-ons like customisable services for configuration, maintenance, update integration, migration from or to other products / product versions.

Thus, the definition of requirements for ICT components and even for complex ICT systems need no longer result in lengthy design and development phases, but can be accompanied by quick though intensive market reviews and tests, which may be followed by quick procurement decisions based fully or largely on commercial products with or without lifetime maintenance.

The advantages both for end users as for system administrators are plentiful. From the user perspective, for example, common place graphical user interfaces facilitate getting used to a new system or to new applications in an existing system, since features present in many applications are accessed more and more often in the same form, so that the user can easily identify and concentrate on new and unknown features to be used.

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

For the administrators, the advantages range from having access to "frequently asked questions" and provider hotlines and thus often well tested solutions helping with day to day problems typical for the product over the better availability of bug fixes for widely spread products as compared to having little or no support for a system that was only devised in one or very few pieces, to the possibility of fully concentrating on user oriented administrative tasks by outsourcing tasks like maintenance, release changes or similar tasks so that they no longer intermingle with the daily routines.

The benefits of outsourcing tasks and services formerly performed within an organisation become obvious when such tasks are only needed from time to time, when equipment needed for these tasks is costly but only infrequently used, when the people responsible for these tasks need special, cost and time intensive training but have little opportunity to use their skills etc. If these tasks and services can be done by outside providers without the need of being familiar with the daily routines of the system or organisation, outsourcing may lead to a less costly and more professional performance of systems.

But even ICT services needed constantly are more and more often subject to outsourcing, as e.g. wide area communications services, customer support (hotline), system administration.

Another example for outsourcing is the operation of systems where, especially in a military environment, high rates for personnel changes are opposed to a long and extensive training required. In such a case, the continuity of systems operation can be better achieved by constant assignment of external specialists.

Other candidates for outsourcing are power supply, water supply, where the label "commercial service" comes into play with the increasing privatisation of these sectors, or services like facility management which may include the employment of private security services.

Paradigm shift in procurement

To profit from the described benefits of using COTS products and commercial services, in many countries the military has already adopted a strategy to use COTS products wherever possible. However, the subsequent changes needed in the procurement processes for new systems as well as for replacing or enhancing existing proprietary systems, sometimes even systems not yet fully operational (and with designed lifetimes of another five or ten years), with COTS products have not yet in all of these countries been fully accomplished.

Another development is that military systems can no longer be easily separated in ICT and non-ICT systems. Electronically interconnecting systems that used to be isolated and only dependent on people to transfer information from one to the other, or introducing and continuously improving "intelligence" in weapon systems via embedded systems, electronic sensors etc., automating logistics, setting up automated chains of interdependent information processes with growing complexity as part of decision support processes allowing, e.g., increasingly real-time situational awareness, are just a few examples showing that information and communications technology has become almost omnipresent in all military systems.

And there is yet another aspect to interconnection and interdependence: the number of systems to which commercial services as e.g. energy supply or wide area communications are indispensable continuously increases, leading to inevitable dependencies of military systems from and interconnection of military systems with systems and services from the civil sector.

Risks by use of COTS products and commercial services

Both the adoption of the maxim of consequent use of COTS products where adequate products are available, and the fact of increased interconnectivity and interdependency within the military and between the military and the non-military sectors lead to new classes of risks.

These classes of risks comprise technical risks as well as risks from organisational, procedural, even political origins, which may, for example, originate from

- System inherent risks due to complexity and heterogeneity of system components, including bugs, backdoors, manipulated chips etc.
- Increasing vulnerability and attack options by interconnecting systems with one another and with commercial open networks (Information Warfare),
- Dependence on products not implemented under military control, which thus have to be operated as "black boxes"
- Dependence on suppliers of equipment and services that operate world wide and whose performance may be unpredictable.
- Political risks when a product is completely or partly produced in a country that is not a friend or partner or changed from friend to adversary
- Risks by using products or integrating products into existing systems that do not meet all the requirements originally formulated for a certain task
- Loss of support and continuity when the manufacturer or a product line with "guaranteed" availability cease to exist

The same range of problems have to be considered when commercial services are used, be it just sporadically or as an essential component of a military system of service. Here, it is also crucial to recognise and take apply appropriate measures against problems that may arise from:

- External personnel having direct or indirect access to military systems, e.g. via direct or remote maintenance,
- external personnel having access to people via social engineering techniques,
- risks introduced by sporadic unavailability of services supposed to "always" available, and
- risks caused by attacks on normally highly reliable services indirectly affecting systems or services depending thereon.

Risk assessment and risk management

In dealing with these problems, the solution cannot be to simply avoid the origins of these risks, e.g. by avoiding the use of COTS products. They have to be accepted as an inevitable side-effects of the need to use COTS products and commercial services, and it has to be acknowledged that these side-effects have to be dealt with. We have to learn to assess and manage these risks as a part of daily life.

To be able to deal with these risks, however, we have to understand that *all* of these risks really have to be recognised and consciously acknowledged as *risks* at all organisational levels.

It is not enough to have IT security experts deal with typical IT security risks, although achieving a high standard of IT security by implementing and managing well tuned and harmonised IT security measures is a fundamental part of successful risk management.

In the October 1999 symposium, I described a threat model and suggested possible procedures (see [1]) for a holistic security management. In the conclusion, I said: " Security management should be designed to effectively assure and support operation of a system (of systems), including all the processes it is designed for. It should be based on a „holistic" view of all security aspects to enhance abilities to detect and correctly assess irregularities and to invoke adequate countermeasures."

Managing security (i.e. managing the measures to achieve the goal) is in this context equivalent to managing risks (i.e. dealing with the problems and keeping them low), where the word "management" indicates not one time actions and static solutions, but continuous analysis of protocols, reviewing the efficiency of technical and organisational measures and procedures, acceptance by the users and so on.

Managing security within an organisation also should be equivalent to enabling secure use of systems and services made available within the organisation

For dealing with risks in connection with the use of COTS products and commercial services, this means that the grade of security (from unsecure to secure) of every specific system or service – as a whole or as a component – has to be assessed and taken into account during the installation or integration by adequate technical and / organisational and /or procedural measures.

Technical aspects

The use of "secure" products, e.g. evaluated along the Common Criteria, is only sometimes a solution, as new releases would have to be re-evaluated and the evaluation process is time and resource consuming, preventing that the latest technology can be made available in a "secure" product at (almost) the same time as the equivalent "normal" product. Also, "secure" products are much more expensive than their "normal" counterparts, which may have a considerable impact on the cost effectiveness and thus means that every day "normal" products have no real alternative.

Technical measures to reduce risks are more and more often based themselves on COTS products and services, e.g. by use of firewalls, anti-virus software, intrusion detection systems, commercial computer emergency response services and so on, where the quality and reliability of these products is very often mainly based on shared positive experience with the product and, for reasons of rapid changes to continuously adapt to new threats, only rarely on evaluation.

However, technical measures may be weakened, if not useless if negligence and carelessness of both users and administrators cannot be considerably reduced. To achieve this, a considerable rise in awareness of the existing risks is required.

Awareness

An overarching risk assessment and subsequent risk management can only be successfully achieved when all parties involved in all stages of the life cycle of ICT systems, i.e. in requiring, designing, deploying, and finally using these systems or the information provided by them, which means more or less everybody, are aware of the imminent problems and willing to take responsibility in the risk management process.

Awareness in this context means

- recognising and acknowledging the existence of a new quality of risks created by the consequent use of COTS products and commercial services

- recognising and acknowledging that these risks have to be dealt with in a co-operative way,
- willingness to contribute to risk reduction according to one's position and tasking
- consequent use of existing security measures
- encouragement of everybody else to do so as well,
- attention to unusual events or obvious security breaches,
- ...

Although these characteristics are independent of whether the systems are pure COTS or using many or few or no COTS components, or of whether they are connected to other systems or to commercial networks, it has to be understood that to cope with the risks induced by increasingly interconnected and interdependent COTS-based ICT systems and direct or indirect use of commercial services, a high level of awareness not just with the security people is a precondition for a successful risk management that enables secure use of these inherently "unsecure" products and services.

This high level of awareness from the simple user through to the highest management level has to be reached step by step, including the broader coverage of security issues in education and training as an integral part of learning how to use and operate a system, supported by a variety of exercises both for crisis management training and for evaluating whether present technical and organisational measures are appropriate to deal with critical events.

A prerequisite for adequate awareness is the availability of comprehensive but easily understandable information on risks, on security measures, on how they work, on possible effects of omitting or ignoring security measures and so on.

Conclusion

For successful risk management, an important prerequisite is to achieve interaction and co-operation between people at all levels: Reports should be encouraged

- of obvious incidents as well as of unusual behaviour – no report should be laughed at or carelessly put aside,
- on events someone has caused himself – helping to reduce or solve a problem should be valued much higher than "finding and punishing the culprit"

but also on successful events such as

- successful integration of "unsecure" products – how to configure them, what sort of extra measures are used,
- new measures that improve early detection of events or increase the number of successfully rejected attacks etc.

Accepting that the use of COTS products and commercial services will continuously increase in the military environment, the obvious benefits have to be levelled with not quite as obvious risks on one hand and, on the other hand, with the possibilities available and duties unavoidable to actively manage these risks.

References

[1] S. Jantsch: "Assessing threats and vulnerabilities", presented at the NATO RTA/IST Symposium "Protecting NATO Information Systems in the 21st Century", Washington D.C., 25.-27.10.1999

C3I Systems acquisition and maintenance in relation to the use of COTS products

S. Rampino
M. Fiorilli
Alenia Marconi Systems SpA
via Tiburtina Km 12,400
00131 Rome, Italy

1. Summary

The paper attempts to highlight the main pros and cons of embedding COTS products in military C3I Systems in the overall framework of Systems Acquisition and Maintenance, basing on Alenia Marconi Systems industrial experience. Significant programs are briefly outlined in this sense, providing the reader an opportunity to consider the issue from the "practical" perspective.

2. Introduction

Military C3I Systems are complex, software intensive Systems, conceived and designed to assist the users in the analysis and solution of operational and management problems, be it tactical or strategic.

Given the functionality required and the dynamic environment in which such Systems are called to operate, C3I Systems can be classified as both Real-time Systems and Information Management Systems.

In Real-time Systems reaction times to external events or to user actions must not only be "fast" but also within accurately predictable limits. Information Management Systems, on the other hand, are conceived to facilitate the job to the user, avoiding repetitive or trivial activities and allowing the operator to focus on the most difficult part of the job: the decision making.

C3I Systems combine all such needs. Data coming from electronic sensors must be collected, filtered and fused. They must be correlated to historical or contextual information, elaborated and synthetically presented to the user; this, in turn, will be given appropriate tools to generate control data and exchange orders and messages to cope promptly and effectively with any situation.

Due to the complexity in terms of functions, of amount and types of data to be dealt with, of inbound and outbound interactions with other Systems in the context of a hostile environment, and more, a C3I system usually is a hard test for the system engineer.

The industrial and operational context in which C3I Systems are conceived, procured and operated is complex as well.

The evolving organisation of modern armies, the contemporary social/political changes, the extreme acceleration of the technological evolution and the increased attention to the cost-effectiveness of C3I systems call for the introduction of new methodologies that combined with modern Standards allow Evolutionary System Development. In order to provide affordable, leading-edge capabilities, Defence Industry is seeking to take advantage of commercial technology wherever possible but by doing so it is changing its role and adapting its competence.

In such a framework, this paper deals with C3I Systems acquisition and maintenance considering AMS specific industrial experiences and projects. A rationale is proposed which explains why Commercial-Off-The-Shelf (COTS) products and technology are increasingly used in C3I Systems, together with potential benefits and risks that need appropriate risk mitigation strategies.

3. International Standards

The increasing availability of standards, applicable to the overall C3I Systems engineering process, came in handy to system designers as immediate, practical solutions to many of their basic problems, such as the need to accommodate operational requirements with financial, industrial and technological constraints.

The definition and wide acceptance of standards, pursued by Industry, Customers Associations and Scientific Communities, ultimately allows a systematic approach to complex problems, leveraging from previous experiences and know-how. Most standards, in addition, are "cook books" for partitioning engineering problems, easing the task to rationalise and optimise solutions.

The definition of standards in the information and communication technologies domain is a dynamic

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

process and it involves many International Entities and Organisations.

Standards exists today for Military and Civil applications which embrace the whole System Life Cycle, from design up to implementation and production, addressing System Engineering, Software Engineering, Quality and so on. Compliant tools and products are readily available right after the standards themselves.

In this framework methodologies and products that, although missing an "official certification", are accepted by users, makers, and designers for their large utilisation, become "standards" as well.

"DE JURE" standards are therefore those established and proposed by appropriate organisations and working-groups. "DE FACTO" standards are those systems, products and methodologies that for their importance in terms of market weight, popularity and potential economical advantages can not be ignored. It is important to acknowledge the existence of both.

We can observe that, in recent years, Military and Civil Standards tend to converge on core principles, so establishing an important link between civil (commercial) and military application domains (see fig. 1).

As an example, MIL-STD499B and IEEE122 are two important Military and Commercial guidelines that focus their attention on the System development process rather than on the object "System".

Both standards promote innovative concepts, such as:

- the need to adapt the "standard" process to the single project and its specific requirements and risks;
- the recurrent applicability of the processes at all levels;
- the iterative use of the processes (evolutionary Systems Management and Development);
- the combined use of traceability techniques and system models to manage projects complexity by linking operational requirements to system solutions;
- the "project database" to keep memory of choices and decisions made along the project;
- the Integrated Product Team concept, gathering all competencies needed for the whole Life Cycle from the project start.

The convergence of military and civil guidelines can be explained with the commonality found in the design of a complex system, be it for commercial or defence application. System designers have always to consider and exploit technologies, methods and products available on the market or off-the-shelf, while behaving in line with good engineering principles:

- manage efficiently the systems development process, using effectively all technical and human resources;
- implement a modular, flexible, expandable, scalable system;
- optimise functionality;
- maximise reliability, survivability and re-use.

As a result, the convergence of Military and Civil standards is producing the first essential step towards the use of COTS in defence systems: the availability of top-quality COTS products and subsystems compliant to military applications.

But more than that, such convergence opens a new frontier as to the "re-use" or even "dual-use" of military C3I systems, components and functionality in the civil domain and vice versa.

4. C3I Systems "Dual-Use" and "Re-Use"

In this framework, "dual-use" of a C3I system is meant as the possibility to use an existing military system, as it is, in a civil application and vice versa.

This possibility is important, as a typical example, in emergency, unpredictable situations when the deployment of military systems constitutes an immediate solution to compensate the inadequacies and deficiencies of civil protection infrastructures. Such infrastructures, being expensive to maintain, are reasonably sized and designed to cope with limited catastrophes. Military C3I Systems, by definition capable to operate in extreme environmental conditions, provide functions directly applicable to the civil domain such as planning, deployment and management of staff and equipment. As an example, an Army Corps C3I system may be used to organize civil or mixed convoys and the set up, management and logistics of military and civil support personnel and infrastructures in dangerous or threatened areas. This may solve critical situations, provided Military Systems are interoperable with their civil counter parts deployed.

"Re-use", on the other hand, is meant as the possibility to build a C3I System by tailoring components or modules of an existing system and by integrating them with newly developed, "ad hoc" ones. The scope of re-use, more and more a practice in defence as well as in civil industry, includes COTS devices and equipment but may also be applied to architectures, tools, design and development methodologies or even just the system developers know-how.

Trying to consider what makes the difference between any two C3I Systems or, as well, what makes them similar, it is natural to note that any complex system of this class is characterised by the following major features:

- The requirements
- The development and design methodologies.
- The functional architecture.
- The physical architecture.
- The HW and SW technologies.
- The specific functions (so called "applications").

In relation to C3I Systems re-use and dual-use it is important to consider the industrial and technological standards available at all these levels, nowadays common to military and civil domains.

Independent researches and studies, for example, on Requirement Analysis, one of the most important phases in C3I System design, demonstrated that any command and control activity may be decomposed in a logical, looped sequence of steps: data acquisition, data processing, situation assessment, planning, plan evaluation, plan execution and back.

As for the development and design methodologies, Evolutionary Development is taking the place of Waterfall Development as the most efficient and cost-effective methodology. Evolutionary Development is an adaptive approach to Design, Development and Maintenance of C3I Systems. The evolutionary life cycle has long been looked at as the key solution to guarantee the necessary flexibility to cope with operational, technological and economical changes that may occur along the life of a C3I System. This approach implies the use of proper methodologies and tools to define and keep track of system requirements, to develop system specifications, to plan and manage its implementation, integration, test and acceptance, up to the provision of logistics support, adaptive and corrective maintenance. It is often associated with the use of Rapid Prototyping tools and techniques combined with high-level 4GL languages and Object Oriented programming paradigms on top of commercial HW and SW development platforms.

The approach to functional architecture of a C3I System is based upon a standard reference model, the Open System model, which responds to requisites of software modularity, scalability and reusability. This model is characterised by a set of functional layers interacting with each other and providing services through specific interfaces. Besides being compliant to the model, a truly "Open System" uses international standards for such interfaces, so that application modules may be ported and still be able to run and operate from an "Open System" to another. Examples of accepted standards exist for:

- User Interfaces (Motif);
- Application Program Interfaces (API);
- RDBMS queries (SQL)
- Graphic (PHIGS, GKS) and Windowing libraries (Windows, X-Window);
- Communication Protocols (ISO/OSI stack);
- Operating Systems (POSIX)

The Evolutionary Life Cycle and the "Open-System" architecture are common solutions both for military and civil C3I Systems. This in turn implies the use of common:

- design and maintenance methodologies
- architectures and functions
- man-machine interface tools

The standard physical architecture for C3I Systems is the Client-Server one, apt to implement internet and intranet Web architectures, with distributed HW and SW processing (LAN/WAN connecting PCs, workstations, peripherals, database and communication servers) running on top of commercial/ standard OS (Unix/Posix, WindowsNT).

Common technologies include digital transmission devices (Ethernet, FDDI), communication protocols and routers (X.500, X.25, TCP-IP/UDP), multiprocessors ADP, ergonomic I/O equipment (monitors, pointing devices, keyboards).

From the functional point of view, a number of standard services, so called "Common User Functions", are required both for military and civil C3I Systems: Data Handling, Communications Handling, Event Handling, Map Handling, System Management, Security. The use of COTS SW is particularly indicated in digital network control, graphical applications and display, data management and distribution.

The application of standards to C3I military and Civil Systems at all such levels is the bridge between the commercial and the military worlds and favours the use of military modules, functionality and tools in civil systems and vice versa. In this framework, industrial competition and shrinking budgets have been pushing the Operational Users and the System Designer to consider and exploit, to the maximum possible level, technologies available on the market. Gradually but steadily, military requirements have been revised in this new light and partly mitigated, thus allowing the use of commercial products, hardware and software, as system components.

5. COTS Technologies

Full-Mil equipment, developed to address specific military applications, has always been expensive due to the need to develop advanced but unique and therefore costly solutions. Furthermore, the development cycle time needed for dedicated military equipment often results in technologies being virtually out of date by the time the equipment enters into service.

Suppliers of systems designed to address commercial markets are able to spread their development costs over a higher number of customers and the availability of the product off the shelf drastically shortens the system lead time.

The use of HW and SW commercial components in military Systems appears therefore the easy way to fulfil requirements with reduced budgets. NATO recommendations, along this line, are indeed to:

- use international standards for which commercial implementations exist
- use common specifications for non-standard system components
- promote co-operation for development of non-standard system components

But, even though the technical differences between a COTS product and a MILSPEC compliant one are reduced by the dominant role of standards, many issues remains to be looked at.

Major military requirements impacting COTS HW performance are related to:

- electromagnetic emission control
- "hostile" environmental conditions
- security
- mobility/transportability

The use of commercially available software is even more complex and it bears all the technical implications of software re-use. Examples of major pre-requisites for software modules to be re-usable are:

- portability: software code has to be independent from the operating system and from the hardware configuration;
- interoperability: the interfaces between the software modules and between software and users (MMI interfaces) must be clearly defined and univocally used;
- flexibility: software modules have to work in different operative conditions maintaining their performances.

But apart from such technical difficulties, Alenia Marconi Systems experience indicates that a number of factors deriving from the use of HW and SW COTS products in military applications do increase project risks, the major ones being:

- The difficulty to evaluate the product. Evaluating a commercial product is difficult even if it is "standard" and "certified" in terms of quality. Detailed documentation of the product is rarely available before the purchase; sometimes product characteristics and performances are poorly documented, not documented or even unknown. Only the effective installation and use of the product may allow an expert system engineer to deeply "understand" the product and its compliance to the requirements. Difficulties may arise not only as a consequence of immaturity of released products but also as a consequence of unpredictable performance in stressful environmental conditions. Evaluation is particularly difficult in terms of Security, Interoperability, Robustness, RAMT and Supportability.

- The difficulty to keep the product under control. A commercial product is a "black box" and can not be tailored to the military application. For example, source code of COTS SW modules is never available. But while it is impossible for the systems integrator to modify a COTS product basic functions and structure, market forces may impose frequent release of upgraded versions of the product or of its components. The rapid response of commercial industry to technological developments may also mean the abrupt discontinuation of products. IT market has a rapid rate of turnover in terms of products (a new generation of equipment, typically, appears every 3 years) and suppliers/makers (small and big companies often grow and go bankrupt). This dynamic world is in contrast with the traditional C3I military Systems life cycle (10 or 20 years) and may have negative consequences in terms of logistic and maintenance costs. Even small changes may impact on the whole System Design with major and unpredictable economical consequences.

- The difficulty to support the product. Technical support provided by the vendor is mostly oriented to the average user. When high technical competence is needed, this may not be readily available. In case the production is discontinued, supportability may not be guaranteed.

- The difficulty to procure the product. Product availability-time can not be really controlled. Once the order is placed the delivery time is hardly guaranteed. Even when the product is delivered, its configuration may not be compliant with the order and the time to fix the supply is sometimes unpredictable.

- The "Secret Costs". The price of a COTS product, providing plenty of maybe unnecessary functions, must not be compared with the cost to develop it from scratch but rather with the cost of developing a module that fulfils the minimum, case specific system requirements, to avoid shooting unnecessarily over the target. Price versus Cost comparison must include guarantees, assistance, maintenance, run-time licenses, upgrading agreements and property rights. Making an accurate estimation of the cost for periodical HW and SW upgrading is difficult. Considering that the price of computing power is continuously dropping and that SW portability costs for an "Open Systems" should not exceed 40% of its acquisition costs, a periodicity of 5 years for system upgrades may be the right balance between costs and state of art performances. A higher upgrading rate may be not convenient in terms of LCC (including logistic support, configuration control and so on).

How to maximise the benefits and minimise the problems associated with COTS based C3I Systems is a critical issue that requires a great deal of engineering analysis and trade-offs along the whole System Life Cycle. The use of commercial components requires a specific and systematic approach to avoid technical and project management problems. System integration of COTS products requires new strategies for negotiation of property rights, estimation of system development and maintenance costs, project planning, risk management. It also requires building sufficient flexibility into procurement contracts. Last but not least, personnel must be trained and proper skills must be developed within the Project Development Team.

6. AMS Spa Land Systems Division Experience

The Land Systems Division (LSD) of Alenia Marconi Systems Spa (AMS) offers services and expertise at all levels and for all phases of C3I systems life cycle, including complex systems design, manufacture, integration and support. This capability matured out of more than 30 years of experience in Battlefield and Air Defence C2 Systems.

LSD policy for C3I Systems acquisition, development and maintenance has long been based on the adoption of the above mentioned international standards and COTS elements have been increasingly used at all levels. In relation to this, two significant recent industrial experiences of AMS LSD are briefly outlined in the following.

CATRIN is the acronym used for a Battlefield Communications and Information System adopted by the Italian Army. The implementation of this tactical system, conceived to provide integrated and automated support up to Corps/Division level, started in the early 90's. The final live evaluation of CATRIN on the field has just been completed and the system is operational. It provides equipment and means in areas where effective co-ordination amongst ground and air friendly forces is required to optimise the employment of sensors, weapons and units, particularly avoiding mutual interference between deployed forces that may lead, in the worst case, to fratricidal casualties.

CATRIN is made of three main subsystems:

- SOTRIN, an integrated telecommunications network and related management functions providing communication services to SORAO and SOATCC subsystems;
- SORAO, providing automated support to battlefield surveillance, target acquisition and correlation, aggregation and distribution of intelligence data.
- SOATCC is the core tactical Air Surveillance, Air Defence Artillery and Army Aviation command and control subsystem

CATRIN was developed through a classic waterfall life cycle based on international and NATO standards available at system design time, such as DOD 2167/A for software development and ISO 7498 for communication protocols.

SORAO and SOATCC, the core CATRIN C2 subsystems, are entirely under the AMS Design Authority. An Open System compliant architecture has been adopted for their design to achieve modularity, flexibility and interoperability with other National and NATO C2 systems. All information is exchanged using NATO standard messages (e.g. Link16 J series and Adat-P3, respectively for "bit oriented" and "character-oriented" messages) through ISO/OSI communication protocols. Commercial protocols have been integrated in the architecture (e.g. X-25 for the WAN; TCP/IP for the LAN) but the selected profile for the higher level of the stack is STAMINA, a military version of the X.400 Electronic Mail civil standard with additional services specifically designed for military message handling. ORBATs have been implemented using a COTS RDBMS (ORACLE) but a Geographical Information System has been developed. Application software packages, coded in ADA, have been designed to achieve a high level of modularity and flexibility. Military operators are able, through user friendly interfaces, even to perform dynamic system reconfiguration IAW the role of the specific Army Corps echelon. The use of such standards and of COTS ADP equipment allows the possible re-use of the CATRIN Command and Control Centres also for civil and/or paramilitary intelligence applications.

CATRIN has, overall, a modular and flexible architecture and it has a high potential for dual-use and re-use. The latest AMS C3I systems, such as C2M, for which the use of standards and COTS elements has been even more pervasive, enhanced such potential.

C2M is a mobile tactical Command and Control Centre developed for the Italian Air Force (IAF) and devoted to support Air Surveillance, Command and Control functions. The system has the capability to be connected to strategic and tactical networks through standard digital and analogue interfaces.

C2M is made of two modules:

- the CCTA Module for Surveillance & Tactical Control
- the CCOA Module for Operational Control & Tactical Command

CCTA and CCOA are housed in shelters containing work stations, computers, control and management facilities for voice and data communications, radios and crypto equipment.

Surveillance functions supported include:

- Surveillance (MRT included)
- Threat Evaluation & Weapon Assignment/Allocation
- Offensive, Defensive and Support Missions Control
- Centralised/De-Centralised SAM control (Hawk, Patriot)
- Italian Air Force radar integration (RAT-31S, RAT-31SL, FPS-117, HR-3000, ATCR-33);
- NATO tactical Links handling (Link-1, Link-11, Link-11b and Link-16);
- RASP generation
- UHF, VHF and HF radios handling.

Operational Control and Tactical Command functions supported include:

- Air Space Management
- Planning and management of defensive and offensive air operations
- Air tasking
- C2 Resources Management (control, allocation and deployment)
- Command Post Exercise (CPX);
- Full data exchange recording / reduction;
- Handling of Messages coming from LINK1, LINK11B, e-mail and a number of external systems (ACCAM, ICC, AOIS, STARGATE WAN connections)

C2M is integrated with the ACCAM and AOIS national networks and it is interoperable with NADGE (through Link1). As for COTS, C2M is based on client-server ADP architecture implemented with rugged HW COTS elements. Other standards and COTS components used along the development are:

- SW life cycle ISO-9001 standard.
- Designer 2000 / Erwin;
- ORACLE RDBMS;
- Network protocols (TCP/IP, SMTP, HTTP), information exchange (e-mail) and tools (Netscape);
- 10/100 Base-F Ethernet LAN;
- Sun Solaris OS;
- Unix/Windows NT portability;

The implementation of C2M is remarkable in that it has been achieved through integration of COTS elements and of the following heterogeneous components:

- an existing mobile C2 system, modified to implement the CCTA Module
- the newly designed CCOA Module and the "CARONTE" SW subsystem, specifically developed by AMS for the programme
- the ICC (Interim CAOC Capability) Software Module released by the NATO C3 Agency
- the GFE STARGATE Software subsystem (the prototyped version of which, developed by IAF, is being industrialised by AMS through reverse engineering activities and delivered back to IAF as a "product")

New Projects, such as ACCS, follow these trends, trying to use more and more COTS hardware platforms, to integrate COTS software with MILSPEC one, to adopt innovative methodologies (e.g. Integrated Product Teams) and to refer to world wide accepted standards for the whole System Life Cycle.

7. Conclusions

The use of international Standards is the way to achieve C3I Systems scalability, modularity, flexibility and interoperability, allowing such Systems to operate with other national and international C3I Systems, in different and stressful operative conditions and for different applications.

Since C3I Military and Civil Systems have been designed following common standards, the use of COTS components has been increasing and dual-use and re-use potentials have been enhanced.

Use of COTS information technology in military systems offers reduced development and support costs, improved interoperability, reduced technological risk, accelerated deployment, and support the evolutionary development concept.

In addition, the continuing trend to use and establish updated technical Standards is pushing modern C3I Systems to be based on COTS products but these are effectively "black boxes" and raise risks and concerns that must be handled properly.

Major Defence Industries, such as AMS, are adding to their system development and manufacturing capabilities of MILSPEC oriented Systems, the skills needed to offer COTS oriented, system of systems integration. This involves evaluation of technologies and products available on the market, together with innovative system design and engineering methodologies. Technical and commercial knowledge is required to determine when a system or a system component is a good candidate for migration toward a COTS approach.

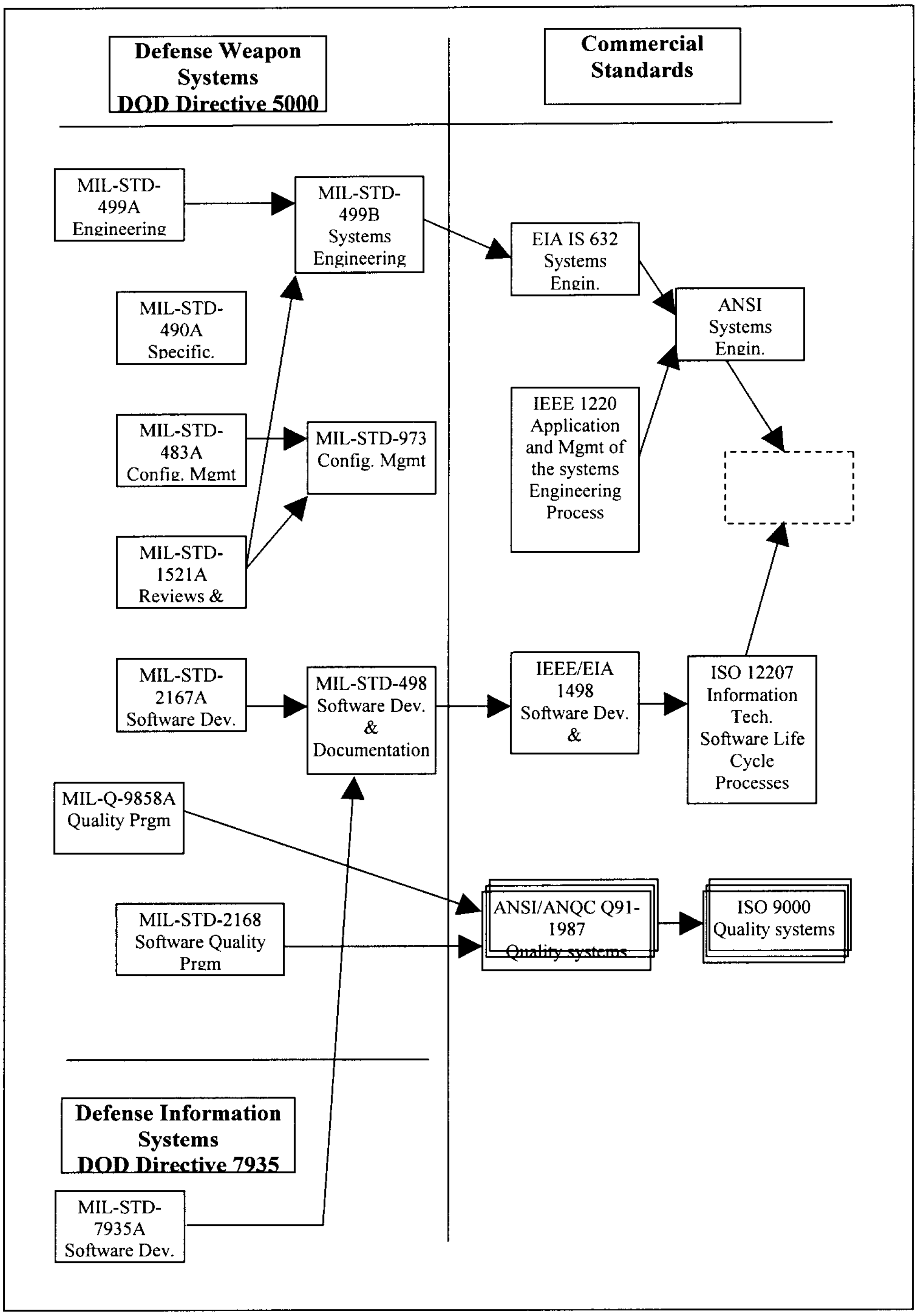

Defense Weapon Systems DOD Directive 5000
Commercial Standards
MIL-STD-499A Engineering
MIL-STD-499B Systems Engineering
EIA IS 632 Systems Engin.
ANSI Systems Engin.
MIL-STD-490A Specific.
MIL-STD-483A Config. Mgmt
MIL-STD-973 Config. Mgmt
IEEE 1220 Application and Mgmt of the systems Engineering Process
MIL-STD-1521A Reviews &
MIL-STD-2167A Software Dev.
MIL-STD-498 Software Dev. & Documentation
IEEE/EIA 1498 Software Dev. &
ISO 12207 Information Tech. Software Life Cycle Processes
MIL-Q-9858A Quality Prgm
ANSI/ANQC Q91-1987 Quality systems
ISO 9000 Quality systems
MIL-STD-2168 Software Quality Prgm
Defense Information Systems DOD Directive 7935
MIL-STD-7935A Software Dev.

Fig.1 Military and Commercial Standard

COTS Software Evaluation Techniques

John C. Dean, CD, B.Sc, M.Math
National Research Council Canada
Software Engineering Group
Building M-50, Montreal Road
Ottawa, Canada K1A 0R6
+1 613 991 6975
John.Dean@nrc.ca

Dr. Mark R. Vigder, PhD
National Research Council Canada
Software Engineering Group
Building M-50, Montreal Road
Ottawa, Canada K1A 0R6
+1 613 991 6972
Mark.Vigder@nrc.ca

ABSTRACT

Employing Commercial Off-the-Shelf (COTS) software products as components in large-scale long-lived systems has been proposed as a way to reduce both implementation and operating cost for the user communities. While this may be the case, the actual benefits have not been confirmed. However, there is factual evidence that some of the suggested cost savings will be offset by the need to address a new set of issues that are raised by the inclusion of COTS components. One of these is the need to evaluate candidates COTS systems early in the development life cycle. Our research is concentrated in the area of physical evaluation of candidate products, that is, actual testing of the products themselves.

The purpose of this paper is to present a discussion of proposed evaluation techniques used to select COTS software components for systems development, to describe appropriate testing techniques for COTS candidates, and to propose an evaluation system which will provide support to ensure timely selection of suitable COTS products.

Keywords
Commercial Off-The-Shelf, COTS, software, evaluation

1 INTRODUCTION

In modern COTS-based systems development we need to evaluate the candidate COTS components at an extremely early stage in the development process. At this stage requirements are generally less than completely defined and often provide only the most general guidance to the evaluator. As with any modern system, the requirements evolve over time. The fundamental difference in a COTS based system is that COTS capabilities have been shown to influence requirements[3,4,10] decisions and thus the evaluation process is inextricably linked to requirements definition.

Some of the proposed COTS evaluation methods have proven to be less than successful because they are based on traditional development paradigms which, while applicable to systems built from first principles, have not been able to easily accommodate COTS software components. Many of these paradigms rely on a highly structured requirements definition and specification that sets the criteria for COTS selection. As such they are slow to react to the fast changing commercial marketplace

Other proposed evaluation processes depend on the pre-qualification of COTS components. With these schemes the developer selects from lists of qualified or certified components which have undergone extensive generic laboratory testing. These components are then incorporated into the current development. The developer must rely on in-context evaluation to ascertain specific knowledge about each candidate COTS software product.

An alternative methodology is one in which the COTS software selection and evaluation influences and is conducted concurrently with the requirement definition process. This approach has advantages in terms of cost and time because it results in a more directed evaluation of components and because it reduces implementation complexity.

2 EVALUATION OF COTS PRODUCTS

Oberndorf et al [15] provide a general background discussion of the issues involved in selecting and evaluating COTS products. In particular, they stress that in-context evaluation is necessary for any reasonable hope of successful evaluation. In context evaluation implies that evaluations are conducted within the scope of the systems to be conducted as opposed to out-of-context evaluation that is conducted against a set of generic criteria.

Current literature provides a number of methods for the evaluation of COTS components. Each of these methods emphasizes one or more critical aspects of COTS software evaluation. This section will discuss highlights of these proposed techniques. This is not meant to be a recommendation as to the validity of these methods, but only an overview. The overriding goal is to identify those aspects of the methodologies that might be useful in developing an integrated approach to evaluation. The information is drawn from a broad range of fields, some of which have different goals than COTS-based systems development, but the information is still pertinent.

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

2.1. COTS-based Integrated System Development (CISD) method

Tran and Liu [16] propose, within the CISD model, a two stage COTS selection process. The first stage is product identification, where candidates are identified and classified. The data for this stage is gathered via vendor documentation, personal experience or other means. The results are a list of potential candidates. The second is evaluation, where the final candidates are chosen (and unsuitable candidates eliminated). In this stage the authors depend on concrete techniques. They state that the COTS evaluation phase requires the extensive use of prototyping techniques. They argue that prototyping is the only way to practically evaluate a COTS candidate within the systems context. They define three critical stages of the evaluation phase; functionality, interoperability, and performance. In the functionality phase the candidates are tested in isolation to confirm that the functionality of the COTS product is applicable to the current application. In the interoperability stage, the candidates are evaluated to ensure their ability to co-exist with other components of the system, both COTS-based and custom developed. The performance evaluation stage consists of a quantitative analysis of the effect of the COTS component on the overall performance of the system.

The final aspect of the methodology is a management evaluation that considers the less tangible aspects of integrating the COTS product. These include such things as training, cost, vendor capability, etc. At the end of this process a final selection of COTS products is made.

The authors also discuss different approaches to evaluation based on constraints such as development time and cost. Two that they highlight are the Comprehensive Evaluation (CE) approach and the First-Fit Evaluation (FE) approach. The result of CE is a list of the most optimal COTS product sets while the result of FE is the first product set which fulfills the requirements. They state that FE is the more cost-effective approach.

Note that this methodology depends on having a relatively complete predefined set of requirements since the product identification stage is dependent on COTS candidates meeting the requirements. The methodology in general is a waterfall-style process in that each stage depends on the results of its predecessor.

2.2. Off-The-Shelf-Option (OTSO)

Kontio et al.[8,9], present a multi-phase approach to COTS selection which begins during requirements solicitation. With their approach the decision to incorporate COTS into the system has been predetermined and thus the OTSO method is only concerned with the actual selection process, not with implementation. The phases are the search phase, the screening and evaluation phase and the analysis phase. In the search phase COTS candidates are identified. At this time the requirements are not fully specified and, in fact, may be quite vague. The screening and evaluation phase narrows the field of potential candidates.

During both these phases the extension of understanding of the product capabilities provides feedback to the requirements definition process. This results in a refinement or modification of known requirements as well as the introduction of new requirements. Evaluations are always performed against a set of evaluation criteria which are established from a number of sources, including the requirements specification, the high level design specification, the project plan, etc.

The final phase of the selection process is the analysis of the results of the evaluation. This leads to the final selection if COTS products for inclusion in the system.

The central theme to the OTSO method is the construction of a "product evaluation criteria hierarchy". This hierarchy serves as a template for situation specific criteria definition.

The conclusions that the authors reach are that criteria definition must be revisited for each project because each project evolves in a different environment at different times. This again implies that evaluation is context-dependent. The OTSO process is iterative because the requirements are both refined and defined throughout the course of the evaluation stage.

2.3. Checklist Driven Software Evaluation Methodology (CDSEM)

Jeanrenaud and Romanazzi[6] present a methodology for evaluating software that employs checklists, which they use to determine a quality metric for each item in the checklist. The process is metric based and provides a numerical result that describes the suitability of the component. This approach is very attractive because it quantifies the evaluation results, however the authors base some of their discussion on the availability of source code and access to individual modules, neither of which are usually available in a COTS product. They also depend heavily on the vendor documentation and demonstrations for supporting data as opposed to in-context, practical evaluation. This may lead to the adoption of unsuitable candidates.

Mcdougall and Squires[13] present arguments why this approach is not necessarily effective as a selection process.

2.4. Procurement-Oriented Requirements Engineering (PORE)

Maiden and Ncube[10,11] propose a template approach to requirements definition that depends on evaluating COTS products. They initially suggest requirements need to be reasonably defined in order to be able to start evaluating COTS products. The process they describe, however, is one in which requirements are defined in parallel with COTS component evaluation and selection.

Within their discussion of lessons learned they highlight that software prototypes are useful in developing knowledge concerning COTS products and their interactions within the overall system. They stress that the selection process needs to proactively evaluate the actual product and not rely exclusively on the vendor-supplied documentation or demonstration.

Although they are directed towards requirements acquisition, the sample templates give a preliminary view of some of the steps needed to perform a justifiable evaluation of candidate COTS applications.

3 TESTING TECHNIQUES FOR COTS EVALUATION

Evaluation of candidate products requires that we adopt some technique to prove the capabilities that interest us. In traditional software development there are two accepted methods of testing software products. They are white box and black box testing. It is not clear that both of these techniques can be applied effectively in the case of COTS software-based systems since both the available documentation and the goals of COTS evaluation are different.

With COTS-based systems there are a number of unique constraints on our ability to conduct effective testing. In general we assume that we have no access to the source code or, in the case where it is available it cannot be modified. This means that we cannot internally instrument the executable. Most vendor documentation that is available consists of user manuals and advertising materials and is not directed at evaluating the operation of the system. For example, it does not describe the behaviour of the system in response to abnormal input. Finally, in COTS-intensive systems much of our use of these products is under non-standard conditions so the testing focus must be skewed towards unique situations.

These constraints influence the goals we are attempting to accomplish with COTS evaluation. Much of our test strategy is directed towards discovery of behaviour under system imposed conditions. We also need to confirm that the product adheres to specifications supplied by the vendor and that it can operate within the system environment, particularly as this pertains to product interoperability. We want to determine if we are able to mask out unwanted functionality as well.

3.1. White Box Testing

White box testing relies on the ability of the tester to examine the internal operations of the software at the source code level. One of the accepted white box testing methods is basis-path testing where an attempt is made to exercise each independent path through a code module. There are a number of interesting ways of determining the independent paths. This type of testing is usually undertaken during actual software development while code is being actively constructed. This corresponds well to the concept of verification testing which confirms that functionality of a system is implemented correctly

3.2. Black Box Testing

Black box testing is designed to allow the tester to treat each code module as a unit which can be defined by its' inputs and outputs (the interfaces to the module) without regard to the route by which an input is transformed into a particular output. With this method visibility into the internal workings of the code module is not necessary and thus the source code is not required. An example of the methods used during black box testing is boundary value analysis where inputs are supplied to the module under test which represent valid, invalid and boundary values. The outputs are then measured and accepted if they fall in the expected output range. The black box type of testing is normally carried out during system integration or after the completion of the coding of a module. This type of testing also is seen during acceptance testing and is considered to be the foundation of validation testing which confirms that the software actually performs the required functions.

The physical testing of COTS candidates is necessarily constrained by the fact that the source code is not available. Some of the testing is for discovery of undocumented features and/or bugs while other testing involves confirming or denying the published vendor data and specifications. Both of these can be seen to be a special case of validation; the first because we are trying to increase our understanding of the candidate under evaluation, and the second to attempt to confirm the vendors claims as to the effectiveness of the COTS product. The various black box techniques seem to be ideal for these purposes since we do not have the visibility into the system that white box testing requires.

Test Methods

One of the recommended methods for evaluating COTS products is to employ scenario-based testing methods. With this method a portfolio of scenarios is created. Note that the scenarios represent typical operating procedures for the system that is to be constructed, not for the COTS product under test. Test procedures are developed based on the scenarios and each candidate is evaluated against the criteria. In this case the initial scenarios are reasonably easily established using the preliminary operational requirements definitions. The results of this type of testing will be confirmation that the qualified candidates perform appropriately in the system context.

Another method that has been suggested by Voas[17] is the use of fault injection techniques. This is particularly effective when access to the internal operations of a product is restricted. The method consists of inserting erroneous values into the data and/or control stream and observing the results. This technique is a good example of evaluating for discovery, that is, to determine unknown or unexpected reactions of the product under evaluation.

4 A PROACTIVE EVALUATION TECHNIQUE

The technique that we have developed during the implementation of our prototype relies neither on a strict requirements definition nor on pre-qualification of COTS products. Rather, it combines the most effective processes of all of the above models.

4.1. The Concept

We begin with a generalized statement of requirements in which we describe only the overall concept of the system to be developed. This initial requirements definition draws much from the operational needs of the users and less from the technical descriptions. We take this approach because we do not, at this early stage, want to eliminate any possible solution to the problem. This allows great flexibility in selection of appropriate COTS software. Only during the detailed evaluation stage do we establish more restrictive technical criteria. Using our prototype system as an example, only after surveying the marketplace for appropriate tools to transfer data, did we select the hypertext transport protocol as the primary transport mechanism.

The next step after gathering initial requirements is to survey the marketplace to determine which candidate COTS components exist that exhibit capabilities compatible with the generalized requirements. We are not attempting to find all the available candidates but only a reasonable selection. Choosing the initial candidates based on generic capabilities somewhat eliminates the competitive aspects of the survey. Should we find a candidate that appears to be an ideal fit we could select that component without further comparison. Our experience has been that we can find one specific component in about ninety percent of the cases without requiring a pre-qualification stage. During the market survey we continue to redefine the requirements based on the knowledge we gain about available products. The information may lead to the addition of or possibly to the removal of requirements.

After choosing the candidates, we analyze existing documentation to determine what advertised capabilities exist that we might require within our proposed system. At this stage we would assume that any of the candidates under consideration could perform adequately in the role. The COTS component is tested to ensure that it indeed performs within its documented parameters. Exceptions are noted but these exceptions do not necessarily eliminate the component. That would occur only if an exception would cause significant harm within our usage context. We do not attempt to assess the undocumented features of a component. This preliminary evaluation is only to determine that the documentation is accurate and that the candidates actually perform as documented.

We then begin a more detailed evaluation of the component by creating a system based test harness and exercising the component within the context of our application. Maiden[10] argues the case for a scenario based testing process as a reasonable and effective testing mechanism. Likely operational scenarios are determined and documented and the candidate is then subjected to operation under the established scenario

Much of this evaluation is conducted using prototype implementation. The actual test suites that are applied to the components are derived from the requirements. Test suites are designed to examine the limits of the product under test. The prototypes are made successively more capable until we are confident that all of the needed functionality of the COTS component has been examined. This evaluation is meant to establish the operating bounds of the component and to enable us to begin to refine our requirements to fall in line with the capabilities of the candidate. We also attempt to determine ways to mask out currently unneeded capabilities.

Finally we define any local enhancements which may be needed to supplement the capabilities of the component. The enhancements are necessary to provide for critical requirements that cannot be implemented using COTS components. These will be implemented in the wrappers and/or the glue code of the system. If a critical requirement is such that an entire subsystem needs to be implemented in-house, that subsystem will be designed and coded so that it can be integrated as a COTS product.

Advantages

The advantages to this approach are significant, particularly in the early stages of development. By restricting the evaluation and testing to the specific needs of the current system we eliminate the direct pre-qualification requirements completely. This allows us to concentrate our efforts on deriving a limited set of tests that exercise only the interesting capabilities of the candidate.

In-context testing ensures that the candidate is suitable for this particular project. The testing is extremely focussed and definitely goal-directed. The actual test cases are designed to exercise only those aspects of the COTS component that will be used for this application. The testing relies heavily on a Black Box approach since the internal operation of the component is usually unknown. Even if we do have access to source code, the goal is to use the COTS software without modification and therefore we must assume that White-Box techniques will not provide useful information. An expanded form of Boundary Value Analysis can supply all the information that we require. This is not to suggest that the testing process is somehow incomplete. We follow the same rigorous approach that we would when planning and implementing testing for a traditional development model but here we emphasize the integration aspects of testing.

We evaluate only existing, current versions, because our evaluation takes place closer to implementation. This ensures that we do not have to repeat the evaluation prior to implementation. We also only evaluate those components that we realistically believe can be used in our system. We select a subset of available components for evaluation by a process of examining candidates only until we feel we have a representative selection. If one candidate appears to be ideal (i.e. it is capable of filling the majority (the 80-20 rule) of the requirements) we do not seek other alternate solutions unless there is some constraint applied. This reduces the number of components that must be evaluated. It also leads to a truly independent assessment of the candidate's capabilities because it reduces the competitive and/or comparative nature of the pre-qualification process.

A further advantage is that we can use the acquired product understanding in future projects to aid in selecting appropriate candidates. We construct our understanding via evolution rather than monolithically. We also gain an understanding of the critical aspects of a candidate COTS product.

Finally this approach fits requirements to COTS capabilities. This leads to a more comprehensive match between the COTS components and the final system requirements. This is not to say that only the capabilities exhibited by the COTS components will appear in the final product; rather, it forces the system integrator to consider carefully whether a particular stated requirement is actually necessary or whether it can be eliminated. Those requirements outside the capability sphere of the candidate COTS products, but deemed necessary, will be met by constructing a component in-house. Implementation of these in-house components follows traditional development processes, except that the component is then integrated into the system in a similar manner to a COTS product. This ensures architecturally consistency throughout the system.

5 CONCLUSIONS

This paper outlines a number of proposed evaluation and selection techniques for choosing appropriate COTS software products for incorporation in large-scale systems. The advantages and disadvantages of each are outlined. We have then proposed a process for the evaluation of COTS software products that takes advantage of the best processes of the different methods as well as introducing new techniques.

Current testing practices as applied to conventional development were examined and their applicability to COTS development highlighted. It is obvious that Black Box techniques are mandatory during in-context evaluation of software but also that the goals of testing are somewhat different from the traditional ones. Scenario based testing provides a good basis for evaluating candidate products. The results obtained from evaluation testing can be used as validation data for system testing.

6 ACKNOWLEDGEMENTS

The research described by this paper has been jointly supported by the Defence Research and Development Branch of the Department of National Defence Canada and the National Research Council Canada.

7 REFERENCES

1. Evan E. Anderson, A Heuristic for Software Evaluation and Selection. Software Practice and Experience, 19(8):707-717, 1989.
2. Jean Bergeron et. al.. Detection of Malicious Code in COTS Software. A Short Survey. In International Software Assurance Certification Conference, ISACC'991998.
3. John C. Dean and Mark R. Vigder. System Implementation Using Off-the-shelf Software. In 9th Annual Software Technology Conference1997.
4. G. Fox and S. Marcom. A Software Development Process for COTS-based Information System Infrastructure. In Fifth International Symposium on Assessment of Software Tools and Technologies, pp133-142, Jun 1997.
5. David Garlan and Robert Allen and John Ockerbloom. Architectural Mismatch or Why it's hard to build systems out of existing parts. In 17th International Conference on Software Engineering, pp179-185, 1995.
6. J. Jeanrenaud and P. Romanazzi. Software Product Evaluation: A Methodological Approach. In Software Quality Management II:Building Software into Quality, pp59-69, 1994.
7. I.M. Klopping and C.F. Bolgiano. Effective evaluation of off-the-shelf microcomputer software. Office Systems Research Journal, 9(1):46-50, 1994.
8. Jyrki Kontio. A Case Study in Applying a Systematic Method for COTS Selection. In 18th International Conference on Software Engineering, pp201-209, 1996.
9. Jyrki Kontio and Gianluigi Caldiera and Victor R. Basili. Defining Factors, Goals and Criteria for Reusable Component Evaluation. In Proceedings of CASCON '96, pp17-28, Nov 1996.
10. Neil A.M. Maiden and Cornelius Ncube and Andrew Moore. Lessons learned during the requirements acquisition for COTS systems. Communications of the ACM, 40(12):21-25, Dec 1997.
11. Neil A. Maiden and Cornelius Ncube. Acquiring COTS Software Selection Requirements. IEEE Software, 15(2):46-56, Mar 1998.
12. Jean Mayrand and François Coallier. System Acquisition Based on Software Product Assessment. In

18th International Conference on Software Engineering, pp210-219, 1996.

13. Anne McDougall and David Squires. A Critical Examination of the Checklist Approach in Software selection. Journal of Educational Computing Research, Vol 12(3):263-274, 1995

14. National Research Council COTS Web Site Available at: http://wwwsel.iit.nrc.ca/projects/COTS/COTSpg.html

15. Patricia Oberndorf and Lisa Brownsword and Ed Morris and Carol Sledge. Workshop on COTS-Based Systems. SEI Special Report CMU/SEI-97-SR-019. 1997.

16. Vu Tran and Dar-Biau Liu. A Risk-Mitigating Model for the Development of Reliable and Maintainable Large-Scale Commercial-Off-The-Shelf Integrated Software Systems. In Proceedings of the 1997 Annual Reliability and Maintainability Symposium, pp361-67, Jan 1997.

17. Jeffrey Voas. Error Propagation Analysis For COTS Systems. Computing and Control Engineering Journal, 8(6):269-72, Dec 1997.

Reliable Tailored-COTS via Independent Verification and Validation

Michael A. Beims
AverStar, Inc.
100 University Drive
Fairmont, WV, USA 26554-8818
mbeims@mail-fair.ivv.nasa.gov

James B. Dabney
AverStar, Inc.
1100 Hercules, Suite 300
Houston, TX, USA 77058
jim@averstar.com

Abstract

An important class of Commercial Off-The-Shelf (COTS) applications is the adaptation of an established COTS product to an operational environment for which it was not originally intended. This tailoring of the established product can provide the expected cost-reduction benefits associated with COTS and still meet system reliability requirements when augmented with an appropriate Independent Verification and Validation (IV&V) activity. We illustrate the tailored-COTS IV&V approach using the integration of a COTS Global Positioning System (GPS) receiver into the Space Shuttle onboard avionics system. The COTS GPS receiver chosen is a proven, reliable navigation aid that has been successfully integrated in numerous military aircraft, ranging from helicopters to jet fighters. However, integration of this COTS receiver into the Space Shuttle avionics system required many changes due to the different avionics hardware environment and the dramatically different flight environment. The key elements of the tailored-COTS IV&V approach are identification of unchanged but operationally affected code, development of automated code analysis tools, software scenario analysis, and exploitation of historical databases.

1 Introduction

Tailored-COTS is an important class of COTS applications in which proven off-the-shelf equipment is adapted to environments for which it was not originally intended. Tailored-COTS can be quite attractive economically, but it presents special challenges to Independent Verification and Validation (IV&V). The integration of the COTS GPS receiver into the Space Shuttle avionics system illustrates typical problems that must be overcome in a tailored-COTS program. The selected GPS receiver is a proven off-the-shelf product that has been successfully integrated into the avionics systems of numerous military aircraft. Changes required for integrating this COTS GPS receiver into the Space Shuttle avionics system include a new interface where a Space Shuttle-specific serial input/output (I/O) card replaced the Mil-Standard 1553 bus Serial I/O interface. Also, the orbital flight environment required significant changes to navigation and satellite vehicle acquisition and tracking algorithms designed for relatively (compared to the Space Shuttle) low speed and low altitude atmospheric flight.

As a result of these significant changes and the high criticality of the Shuttle navigation system, NASA's Independent Verification and Validation facility was tasked to perform IV&V on the Shuttle's modified COTS GPS receiver, specifically the embedded software in the receiver. This IV&V effort required the development of a new tailored-COTS IV&V process that has been very successful. This new IV&V process was based on IV&V techniques employed successfully on traditional mission-critical software development projects. The tailored-COTS environment presents significant new issues in resource allocation and verification and validation techniques.

The paper briefly describes the hardware and environmental differences between the COTS GPS receiver's environment and the Shuttle, and explains the unique issues posed by IV&V of tailored-COTS products. This paper also identifies several IV&V techniques that were successfully used during IV&V of the modified COTS GPS receiver's embedded software. Finally, it presents conclusions and suggests future improvements to the process.

2 Background

The Space Shuttle is a unique aerospace vehicle in that it must operate as a rocket (during launch and ascent), as a satellite (during orbit), and as an aircraft (during entry and landing). These distinctly different flight regimes each present different navigation problems. The current Shuttle navigation system uses star tracker and ground radar for on-orbit navigation and tactical air navigation (TACAN) and microwave scanning beam landing system (MSBLS) during entry and landing. TACAN is a ground-based military enroute navigation system that is being replaced by GPS on all United States military aircraft. Therefore, within a few years, it will be necessary for NASA to replace the Shuttle TACAN system with GPS or to maintain the TACAN ground stations at NASA expense.

The selected COTS GPS receiver was designed and tested for use in military aircraft ranging from helicopters to supersonic jet fighter aircraft. It has proven to be an extremely reliable aid to navigation. Since the selected off-the-shelf unit is a military GPS

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

receiver, it is equipped with the necessary circuitry to allow it to use the precise positioning service (PPS). This PPS capability provides increased accuracy over typical civilian GPS receivers, and reduces vulnerability to radio interference. All of these attributes are desirable for a Space Shuttle navigation system. Since developing a completely new GPS receiver for the Shuttle would be prohibitively expensive and the COTS GPS receiver has these desirable attributes, it was selected as the basis of a GPS receiver for the Shuttle.

Although the selected COTS GPS receiver is a proven, reliable product, there are still many differences between a typical military GPS application and the Space Shuttle. These differences include both the avionics environment and the flight environment. We will discuss each next.

2.1 Avionics Environment

Previous applications of the selected COTS GPS receiver provided control and user interface to the receiver through a control display unit or through the Mil-Standard 1553 bus. An interface manager function in the receiver accommodates the different interfaces, including service specific (Army, Navy, Air Force) variations in the 1553 bus controls. The Space Shuttle uses a modulator/demodulator (MDM) serial I/O bus, which requires a new hardware interface in the receiver and also new interface software in the receiver. Additional interface software changes inside the receiver were needed to process Space Shuttle flight software unique antenna lever arm and attitude references.

2.2 Flight Environment

There are several differences between the Shuttle flight environment and military aircraft. These include vehicle speed, altitude, and flight attitude. Although the original motivation for installing GPS in the Shuttle was replacement of TACAN (available only during the landing phase), GPS is available during all Shuttle mission phases. So the Shuttle avionics system was modified to use GPS in all flight phases, including the launch and orbit phases in addition to the landing phase.

Speed is a difference in the flight environment as typical speeds for military aircraft range from zero in hovering helicopters to less than Mach 3 for jet fighter aircraft. This contrasts with the Space Shuttle, which on orbit operates at speeds of up to Mach 25. Furthermore, navigation calculations for military aircraft are typically performed using either rhumb line or great circle techniques. Except during the landing approach, the Shuttle must use ballistic propagation algorithms.

A second consequence of the Shuttle's high speed is that satellites are typically visible for a much shorter period, thus increasing the satellite selection workload. For an aircraft, a satellite is typically visible for approximately six hours as the satellite traverses from horizon to horizon. For the Space Shuttle, this visibility window is reduced to approximately 45 minutes. Satellite selection in the COTS GPS receiver requires constantly choosing from among all the visible GPS satellites the set of four satellites that provides the best navigation solution. This is a complex calculation, and since the high speed of the Shuttle requires more frequent satellite selection computations, the computational resources available for other tasks are reduced.

Altitude presents another difference in the flight environment, since military aircraft typically fly at altitudes of less than 20 kilometers while the Space Shuttle flies at altitudes in excess of 500 kilometers. An important consequence of the Space Shuttle's increased altitude is that at any moment, more satellites can be visible to the Space Shuttle than are visible to an aircraft in atmospheric flight. This increases the number of satellites that must be evaluated for inclusion in the navigation solution, further increasing computational workload. Additionally, on orbit, the Space Shuttle has line-of-sight visibility to GPS satellites up to 20 degrees below the local level plane, potentially changing parameters of the satellite selection algorithms.

A final flight environmental difference is vehicle attitude. Military aircraft, including jet fighters, spend most of the time in a heads-up attitude. Therefore, for military aircraft a single GPS antenna on an upper surface has unobstructed line-of-sight to a sufficient number of GPS satellites most of the time. The Space Shuttle, on the other hand, frequently orbits in a heads-down attitude for extended periods. Also, during entry, the Shuttle flies at a relatively high pitch attitude, which obstructs line-of-sight to a large portion of the sky. Consequently, the Shuttle must use two GPS antennas, one on an upper surface and one on a lower surface. Since the navigation algorithms determine position based on the location of the receiving antenna, it is necessary for the software to decide which antenna is receiving the signal from each satellite, a problem not faced by the COTS GPS receiver.

2.3 Similarities to Other Applications

The differences between avionics and flight environments just described are significant and extensive. However, most of the COTS GPS receiver hardware and software were compatible with the Shuttle environment. For example, the basic hardware characteristics such as packaging and power required no change. Much of the COTS GPS receiver's internal software also required no changes including the radio frequency control processing, including the internal receiver moding and control, and the geometric calculations to reduce geometric dilution of precision (GDOP). Other unchanged off-the-shelf functions of particular importance are the military performance accuracy and the security related processing in the receiver (Selective Availability, anti-spoofing, and antijamming). As these unchanged characteristics far exceed the new and changed characteristics; it is reasonable to treat the Shuttle's modified COTS GPS receiver as tailored-COTS rather than as an entirely new product.

3 Approach

The tailored-COTS environment presents significant new issues in resource allocation and verification and validation techniques. Other researchers have documented similar modifications to their processes for COTS applications. These modifications include process changes running through the entire range of the Procurement, System Engineering and Integration activities and have been documented for United States military procurements. Software engineering processes must be tailored to incorporate new computing system standards and methodologies. Avionics System Engineering processes must evolve and adapt to dynamically changing COTS Non-Developmental Item product lines that incorporate emerging standards [11]. While the solutions provided are employing commercial standards and off-the-shelf products, a major role to be played by the integrating organization is to become the trusted subsystem integrator. The organization will put wrappers around the commercial technologies to meet the customers' needs [17].

The first consideration in any IV&V effort is to determine the optimum allocation of finite IV&V resources. This process is complicated in the case of tailored-COTS because it is neither necessary nor economically feasible to perform comprehensive IV&V of the entire software product.

The software in a tailored-COTS product can be partitioned into three classes: new or modified, not modified but affected operationally, and unaffected. The first class, new or modified, is easy to assess since it clearly merits IV&V and can be dealt with using standard IV&V methods. The third class, unaffected, is also easy to assess, as it clearly does not merit IV&V. But the second class, not modified but affected operationally, presents two problems: identification and verification. The focus of this paper is the development and application of methods for identifying and verifying software code of the second class.

3.1 Criticality Analysis and Risk Assessment

A fundamental step in any IV&V project is the allocation of the available technical staff resource. Both the number of analysts and the overall project schedule constrain the activity. Since the amount of potential IV&V work on any complex project exceeds the available resources, it is necessary to allocate the resources to achieve the greatest benefit. NASA's IV&V contractor on this modified COTS GPS receiver project, AverStar, Inc., employs a process known as Criticality Analysis and Risk Assessment (CARA) to guide this resource allocation [12].

3.1.1 CARA Overview

CARA is based on the notion that there are two key factors to consider in IV&V resource allocation: criticality and risk. Here, criticality is a measure of the consequences of an error in a particular software function. Risk is a measure of the likelihood of an error. Table 1 provides a synopsis of the CARA process.

Table 1 : Criticality Analysis and Risk Assessment Process

Phase	Step	Activity
Preparation	1	Establish CARA team including domain experts and IV&V process experts.
	2	Decompose the software system into critical functions. These should be functionally distinct and sufficiently small to permit analysis by a single individual.
Evaluation	3	Develop criticality and risk criteria, starting with the baseline CARA factors.
	4	Rate each critical function using the selected criteria and compute overall CARA scores.
IV&V Scoping	5	Set threshold levels to map an IV&V level (degree of scrutiny) to the CARA scores.
	6	Perform software size estimates using measures such as source lines of code or function points.
	7	Estimate IV&V effort required using the size estimates of Step 6 and IV&V levels of Step 5.
	8	Repeat Steps 5 and 7 as necessary such that a feasible work plan is achieved.

CARA is an iterative process. It is performed once at the outset of an IV&V project, then repeated periodically. This iteration is necessitated by several factors. For example, as the project progresses, the IV&V team gains greater insight, enabling refinement of the analysis. Also, the software requirements and design can evolve, changing both criticality and risk, and even introducing new critical functions.

3.1.2 Shuttle COTS GPS Receiver CARA

For the Shuttle's modified COTS GPS receiver project, an initial CARA was performed after the IV&V team reviewed all available documentation. This included requirements and design documentation for the baseline military COTS GPS receiver and proposed changes to the COTS GPS receiver's embedded software to adapt it to the Shuttle. The team also reviewed applicable changes to the Shuttle general-purpose computer flight software. Additionally, the team analyzed development flight test data and operational requirements.

The tailored-COTS nature of the COTS GPS receiver IV&V project changed the CARA process significantly. Added factors in assessing risk were necessary to properly attribute risk reduction due to the shelf life of the COTS code. So it was necessary to identify and consider separately the changed and new code and the code that was not changed (or at least not changed much). Other new considerations that affected both criticality and risk were the different operational environment and the availability of historical data.

The differences between the operational environment of the Shuttle and previous applications of the selected COTS GPS receiver affected both criticality and risk. For example, the more rapid change in the relative configuration of the satellite constellation could amplify the consequences of errors in satellite selection algorithms. The differences in operational environment also increased the risk of problems in satellite selection because the algorithms must operate more frequently and track a larger number of satellites.

Risk analysis was expanded to include assessment of the degree to which each unchanged (or little-changed) critical function interacted with new or extensively changed critical functions. This determination was based on analysis of the software requirements and design documentation as well as mission analysis.

Risk analysis was augmented via problem databases maintained by the manufacturer and the United States Department of Defense. The reasoning was that critical functions, which had historically experienced a larger number of programming and operational errors, were considered more likely to contain errors with respect to the new environment.

Prior to the initiation of the IV&V effort, NASA had flown a prototype modified COTS GPS receiver on several Shuttle missions. These flight experiments provided a wealth of data that the IV&V team analyzed to gain further insight to aid the CARA.

The initial CARA guided the detailed requirements analysis phase of the modified COTS GPS receiver IV&V project. Subsequent CARAs were augmented with the lessons learned in previous IV&V phases, additional flight experiments, and continued monitoring of the operational experiences of military users of the selected COTS GPS receiver.

3.2 Tools

Software analysis tools are valuable in any IV&V effort because the tools can automate certain analysis tasks. Software tools are especially useful in the case of tailored-COTS because the majority of the software already exists when the project begins, so the tools can be used much earlier in the IV&V activity.

Many standard reverse engineering and software analysis tools are useful aids to IV&V. Among these are commercial tools intended to support maintenance of code [16] and various tools in an advanced state of research. For example, research tools exist that compute worst case execution time and that handle advanced programming constructions including: limited recursion, analytically complex loops with multiple exits, non-looping functions, function pointer calls, data pointers, non-terminating loops and functions, and multiple entry points [4]. Other useful tools produce diagrams to aid understanding and document the design [18]. Tools that compute cyclomatic complexity are also useful, particularly in support of the CARA, as cyclomatic complexity has been shown to be a reliable risk indicator [6].

Several static analysis tools are especially useful in identifying code that interacts extensively with new or changed code [4, 5, 15, and 21]. Set/use identification tools allow an analyst to rapidly assess the interactions from a data flow perspective. Flow chart generators and call trees provide a control flow perspective. Of course, these tools are also valuable during detailed analysis of the critical functions selected via the CARA.

Another class of tools that is particularly useful in the tailored-COTS environment is special purpose code audit tools. These are tools designed to automatically locate and assess particular patterns. For example, while on orbit, the Shuttle has line-of-sight visibility to more satellites than does a typical COTS GPS receiver user in atmospheric flight. Therefore, it was necessary to verify that all applicable tables and arrays are properly sized for the Shuttle environment. This task was well suited to a custom code analysis tool. Special purpose audit tools were also produced to rapidly locate additional instances of problems identified from historical databases. Among these were tools to identify and check instances of function calls, to search for potential instances of division by zero, and to search for potential instances of indexing arrays beyond their limits.

3.3 Scenario Analysis

Software scenario analysis is a team problem solving technique that seeks to understand the behavior of a software system responding to various external events. A software scenario begins with an external event, and ends when the system resumes nominal cyclic operation or an error occurs. A similar team approach has been used to verify requirements for real time spacecraft systems [19] and relates to techniques for stepwise refinement and verification used in the Cleanroom approach [13].

Our approach to software scenario analysis can be summarized as the following sequence of activities:

- Using group-brainstorming techniques, a large number of potential scenarios are postulated. This is aided by both operational environment expertise and critical function expertise that analysts have gained in earlier phases of the IV&V project, particularly requirements analysis.
- Using a process similar to CARA, all scenarios are ranked based on criticality and risk.
- The primary IV&V analyst assigned to the critical function most involved in the scenario initiates analysis for each scenario. The analyst formally documents the control and data flows in a scenario analysis report.
- When flow passes to another critical function, analysis responsibility is transferred to the analyst with appropriate critical function expertise. This transfer is repeated until the scenario reaches a logical conclusion. Each analyst records his or her findings in the scenario analysis report.
- The lead analyst for the scenario presents the report at a peer review meeting and the entire scenario is discussed in detail. This step verifies the results and often suggests new scenarios and interactions with other critical functions.

Operational scenario analysis is frequently a valuable IV&V technique. But, it is particularly useful in the tailored-COTS environment because it is an efficient means to identify and evaluate the behavior of critical functions that are not changed but that are operationally affected by changes in other areas. In the case of the modified COTS GPS receiver, operational scenario analysis resulted in the identification of a number of subtle software issues. Additionally, operational scenario analysis was valuable in follow-on CARA updates and resulted in the inclusion of two new critical functions in the IV&V activity.

3.4 Model Checking

Model Checking is a formal verification technique in which assertions about a finite state machine process model are automatically tested [1– 3, 7 – 10, 15, 22, and 23]. Model checking is useful for a variety of verification approaches [20]. For example, it is useful as a means to assess liveness properties of the underlying finite state machine [8]. Model checking has also been demonstrated as means to automatically generate test cases [2].

The principal difficulty in model checking, from the analyst's perspective, is producing the model. It is necessary both to develop the model and to verify its equivalence to the system under consideration. Tailored-COTS can be an ideal candidate for model checking because the majority of the source code exists when IV&V begins. Consequently, it may be possible to automatically translate the source code into the modeling language, reducing labor and increasing the likelihood of an accurate model.

For the modified COTS GPS receiver IV&V project, model checking proved to be an extremely valuable adjunct to the scenario analysis process. For example, a critical portion of the COTS GPS receiver software (Receiver Manager) is implemented as a set of finite state machines. This critical function manages the five satellite tracking channels, which perform multiple tasks. The CARA suggested that this function was high in criticality and risk, and preliminary scenario analysis supported the CARA. Scenario analysis brainstorming revealed numerous scenarios with respect to Receiver Manager. Unfortunately, the complexity of the function would make manual analysis of all the scenarios prohibitively time consuming.

Since the source code was structured as a set of finite state machines, it was a straightforward task to translate the source code into the model checking language Promela [14] for use with the Spin model checker. Using Spin, it was possible to automatically check all of the Receiver Manager scenarios [1]. This allowed us to verify liveness properties of all of the possible configurations of the finite state machine. In particular, it identified a singular situation in which a receiver channel could be frozen in a certain state (a deadlock). Additionally, a byproduct of the model checking process is a scenario trace that shows how the deadlock state can be reached. This information greatly facilitated manual verification of the problem scenario.

3.5 Historical Databases

A major benefit of tailored-COTS is that it has an operational experience base. Insight into operational experience can greatly facilitate CARA and can also help to identify the unchanged but operationally affected code. Some of the sources of operational experience information are:

- User group databases. Since the COTS GPS receiver is a military product shared by all branches of the armed services, there is a joint program office that maintains valuable data. There are often USENET users' groups that can be significant sources of operational information.
- Vendor problem databases. These databases provide insight into both criticality and risk. In some cases, they may even contain useful information on previous tailoring of the COTS product. The COTS GPS receiver manufacturer maintains a problem database that was extremely beneficial to the IV&V effort.
- Test results. There should be a wealth of useful test results for any operational product. This information can augment the problem databases. However, because of its size, it should not be used as a primary reference. In the case of the modified COTS GPS receiver, several Shuttle missions gathered data using different versions of the receiver, including production prototypes.

4 Conclusions and Future Work

The Space Shuttle's modified COTS GPS receiver IV&V activity has demonstrated that COTS can be successfully tailored to operational environments for which it was not originally intended. The key difference between tailored-COTS IV&V and traditional IV&V is the need to identify and verify portions of the software that are not changed but that are operationally affected by the new environment. The techniques that proved most beneficial were a modified criticality analysis and risk assessment process, custom source code analysis tools, software scenario analysis, and model checking. Finally, historical databases were found extremely valuable sources of information.

There are significant opportunities for further research in the area of tailored-COTS IV&V. For example, tools that automatically extract finite state machine models from procedural language source code would facilitate model checking. There is also a need for tools to support the scenario analysis process and to support CARA.

5 Acknowledgements

The authors wish to acknowledge the support of AverStar, Inc. and the NASA IV&V facility in Fairmont, West Virginia. In particular, Prof. Jack Callahan of West Virginia University and Steve Husty of AverStar contributed extensively to the model checking activity. We also wish to acknowledge the members of the modified COTS GPS receiver IV&V team: John Bradbury, Reid Brockway, Don English, Larry Wiederholt, and David Wirkkala.

References

[1] Beims, M., and Callahan, J. *Independent validation and verification of firmware.* NASA 2nd Annual Workshop on Risk Management (WoRM 99). October 28-29,1999. Fairmont, WV.

[2] Callahan, J. *Model checking as a test case generator.* Work in Progress Presentation. Fall 1998. NASA Software IV&V Facility, Fairmont, WV.

[3] Clarke, E., and Gluch, D. *History of model checking.* SEI/CMU Site Visit Presentation. Winter 1998. NASA Software IV&V Facility, Fairmont, WV.

[4] Engblom, J. *Static properties of commercial embedded real-time programs, and their implication for worst-case execution time analysis. In:* Proceedings of the Fifth IEEE Real-Time Technology and Applications. 1999. pp 46 –55.

[5] Hayman, K. *An analysis of ordnance software using the MALPAS tools. In:* Proceedings of the Fifth IEEE on COMPASS '90, Systems Integrity, Software Safety, and Process Security. 1990. pp 86 - 94.

[6] Heimann, D. *CATS-an automated user interface for software development and testing. In:* IEEE Proceedings Annual Reliability and Maintainability Symposium. 1996. pp 163 - 166.

[7] Holzmann, G. *The spin model checker.* IEEE Trans. on Software Engineering, Vol. 23, No. 5. May 1997. pp 279 - 295.

[8] Holzmann, G., and Peled, D. *An improvement in formal verification.* Proceedings FORTE 1994 Conference, Bern, Switzerland.

[9] Husty, S. *An automated software maintenance process for the firmware using model checking techniques draft version.* Master of Science Project Report. University of West Virginia. 1999.

[10] Joseph, S. *Fault injection with model checking,* Ph.D. Thesis, University of West Virginia. December 1998.

[11] Kuehl, C. *A process direction for common avionics developments using commercial hardware and software components: The avionics systems engineering challenge. In:* Proceedings of the 16th Digital Avionics Systems Conference (DASC)., conference publication AIAA/IEEE Volume: 2. 1997. pp 6.4 -1- 6.4-9.

[12] McCaugherty, D. *The Criticality and Risk Assessment (CARA) method.* Workshop on Risk Management (WoRM) 98. 26 October 1998. http://research.ivv.nasa.gov/worm98/proceedings/mccaugherty.pdf

[13] Mills, H. S*tepwise refinement and verification in box-structured systems.* Cleanroom Software Engineering: A Reader. 1996. pp 169 - 197.

[14] Nagulakonda, V. *Creating models from source code.* Work in Progress Presentation. Winter 1998. NASA Software IV&V Facility, Fairmont, WV.

[15] Ogasawara, H., Aizawa, M., and Yamada, A., *Experiences with program static analysis. In:* Proceedings of the Fifth IEEE Software Metrics Symposium. 1998. pp 109 - 112.

[16] Oman, P., Novobilski, A., Rajlich, V., Harband, J., McCabe, T., Cross, J., Vanek, L., Davis, L., Gallagher, K., and Wilde, N. *Maintenance tools.* IEEE Software Volume: 7. 3 May 1990. pp 59 - 65.

[17] Perry, H. *The application of commercial processing technologies to the airborne military environment. In:* Proceedings of the 17th Digital Avionics Systems Conference (DASC)conference publication AIAA/IEEE/SAE Volume: 2. 1998. pp G35/1 - G35/8.

[18] Pierce, R., Ayache, S., Ward, R., Stevens, J., Clifton, H., and Galle, J. *Capturing and verifying performance requirements for hard real time systems. In:* Ada-Europe International Conference on Reliable Software Technologies conference publication Lecture Notes in Computer Science. 1997. pp 137 - 160.

[19] Prywes, N., Rehmet, P., Sokolsky, O., and Lee, I. *Retrospective exploration of safety properties in real-time concurrent systems. In:* Proceedings of the 16th Digital Avionics Systems Conference (DASC), conference publication AIAA/IEEE Volume: 1. 1997. pp 1.1 - 43 - 1.1 - 51.

[20] Schneider, F. *Verification and validation through model checking.* Jet Propulsion Laboratory. California Institute of Technology, Pasadena, CA. October 12, 1997.

[21] Thornley, J. S*tatic analysis and diversity in the software development process – Experiences with the use of SPARK. In:* Ada-Europe International Conference on Reliable Software Technologies, conference publication Lecture Notes in Computer Science. 1997. pp 266 - 277.

[22] Vijayakumar, S. *Use of historical data in software cost estimation.* Computing & Control Engineering Journal Volume: 8 3. June 1997. pp 113 - 119.

[23] Williams, C. *Spin modeling of CLCS redundancy management.* Work in Progress Presentation. Winter 1998. NASA Software IV&V Facility, Fairmont, WV.

COTS Software Supplier Identification and Evaluation

April 2000

Ann Miller
Cynthia Tang Missouri Distinguished Professor of Computer Engineering
Department of Electrical and Computer Engineering
University of Missouri – Rolla
1870 Miner Circle
Rolla, MO 65409 USA

Abstract. There has been a consistent trend to field increasingly large systems. Largeness requires a longer development cycle that is in direct conflict with the need to field systems quickly. Several approaches have been developed to reduce time-to-market. One of the most notable methods in reaction to time-to-field pressures is the inclusion of Commercial-Off-the-Shelf (COTS) as well as Government-off-the-Shelf (GOTS) software packages to perform some of the functions of these new "mega-systems". This paper addresses some of the advantages and pitfalls of the inclusion of COTS components and discusses the need for an evaluation not only of the COTS component but also of the COTS supplier. The paper concludes with some of the lessons learned from the use of COTS incorporation and of supplier assessments over a ten-year span of commercial and government acquisitions.

Keywords. Large project development, COTS components, software acquisition strategy, software supplier evaluation.

Introduction. Large-scale systems are becoming increasingly common, both in military and commercial systems. As systems provide more features and functions, the size of the delivered software increases as well. Sheer size, whether measured in thousands of lines of code (KLOC) or in bytes of program code, is one metric by which to gauge the "largeness" of a system. Measured by size, software content in systems seems to be following a software variant of Moore's Law [1] with exponential increases in size every generation, or approximately every 18 months if the systems are not related by product line. Another indicator of growth is development team size. Complexity and function point metrics are other possible indicators. Some projects have also used pages of documentation as a metric; the author recalls one project in the mid-1980s for which the requirements specification documents in ring-bound notebooks spanned more than six linear feet of shelf space. Whatever measure is chosen for the yardstick, numerous examples of large-scale systems can be found.

As with any task, scale has its effect on software and systems development. An individual can assemble an ultralight plane from a kit; so, too, can one individual design, code, and test a small software program. But the ultralight builder cannot undertake the sole design, development, manufacture, and assembly of a Boeing 777 aircraft. Neither can one software engineer undertake the sole design, development, code, and test of a large-scale software program. In addition to the sheer length of time for such an undertaking, fielding of a large program requires a multitude of skills, as does the assembly of the 777 jet. Thus, a large team is necessary.

Typically, no single organization has all of the expertise to bring a large-scale product to market; even if the expertise were present, it might be unrealistic to apply all of the organization's resources to one product. Thus, development of a large system will likely include suppliers for portions of the hardware and/or software. In addition, there is typically a geographic disbursement not only between the development organization and the software

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

suppliers, but also among the various development teams.

This paper will discuss incorporation of COTS components into large-scale development efforts and will provide some lessons learned from more than a decade of technical contribution on and management oversight of large programs.

Why is this topic important? Capers Jones [2] has summarized it most succinctly: "Software package acquisition actually delivers more software to business and government users than almost any kind of development activity. Yet in spite of the huge volumes of software purchased or leased every year by companies, civilian government agencies, and military services, the process of acquiring packages is curiously amateurish and unprofessional. Some organizations have no formal methodologies for package evaluations and acquisition."

This paper describes a formal software supplier identification and evaluation process that began in 1990 and that has evolved over the years. Specifically, we will examine the need for a structured evaluation of the COTS vendor as well as an evaluation of the COTS product itself. This is not a research paper; it is presented to practitioners by a practitioner. No theorems will be proven, no formal assertions will be derived, no names will be named. However, examples from commercial and government acquisitions will be discussed and references will be provided which span a large spectrum devoted to the topic of including COTS packages in large-scale software development.

The Effects of Scale and Time-to-Market. A large team brings its own set of problems, and the effects of scale on a project have been discussed in the literature, from communication nodes to function points to general project management. One of the most insightful articles concerning scale discussed organizational and management aspects of large projects in terms of the Tower of Babel [3]. The criticality of architecture [4] and testing [5] of large systems has also been analyzed from the practitioner's viewpoint. Life-cycle models for the large systems have been discussed and have evolved over time. [6, 7, 8, 9]. Lastly, there have been numerous papers devoted to development processes for large-scale systems, with many of these based on the Software Engineering Institute's (SEI) Software Capability Maturity Model [10]. Lastly, both commercial [11] and military software acquisition standards have emerged for summarizing best practices. But no matter which life-cycle model and development process and methodology adopted, the basic effect of largeness on a product is that it takes longer to build.

On the other hand, in industry, time-to-market considerations foster rapid fielding of systems, exactly opposite to the effect of scale on a project. Military applications have similar pressures. Once new technology is available to the warfighter, there is the strong desire to ruggedize the hardware component and distribute that technology. Both commercial development organization and military acquisition offices have sought ways out of this conflicting situation. A common approach to field large systems in a timely manner is inclusion of COTS packages to provide portions of the total system functionality. Large-scale government systems may contain GOTS packages as well.

Reasons for COTS Components. By incorporating COTS components in a larger system, the development time for that functionality is decreased; however, such a practice is not a panacea. Just because a portion of coding has been eliminated, the remaining phases of analysis, design, integration testing and acceptance testing remain. Still, there are many reasons to consider COTS packages in addition to the savings on development time: (i) the particular packages might require a specific domain expertise which does not reside in the development organization, (ii) the package may be a de-facto standard which customers expect to be part of the total system, or (iii) it might be an existing product from another part of the development organization which is being reused as part of a business plan to enter new markets.

The Software Acquisition Strategy. The first issue related to any large-scale effort is to determine the acquisition strategy. Acquisition examines three distinct questions. First, what portions of the total system already exist, either through re-used software or through a COTS component? Second, which portions must be developed? Lastly, of those portions that must be developed, which can be subcontracted and which should be developed in-house? In-house development is often reserved for the "family jewels", that is, those portions of the final product that would give the developing organization a competitive edge in the marketplace.

In 1990, the author was the Chief Software Engineer for a large commercial project in which the software acquisition priority was to seek COTS components for as much as possible beyond those aspects which had been identified as required for internal development due to competitive advantage. If COTS packages could not be found, then and only then, would we seek qualified suppliers to generate the remaining functionality. In typical systems development projects, an enterprise will develop most of the product themselves. However, for this product, the initial software size estimate was 12 Million Lines of Code (MLOC) and the critical proprietary code was estimated to form approximately 8% of that total. Thus, with more than 10 MLOC to be subcontracted, it made good business sense to generate a software supplier identification and evaluation plan.

Identification of COTS Components. Once the decision to procure a component has been made, a careful market analysis of potential packages must be made. There are several concerns related to identification of such software packages. The first issue is functionality. A COTS package will most likely not be an exact fit; that is, it may not have all of the required features or it may have additional, unwanted features relative to the system requirements. Since the large-scale system developer probably will not have access to the source code of the COTS package, can the developer assure that the package performs its intended functions and that unwanted features won't be able to be invoked when integrated into the whole? Compatibility is another issue; the COTS package itself will most likely be evolving. What is the vendor's release plan? Typically, a customer incorporating a COTS component does not have control or influence in the package's evolution. Will future releases be backward compatible? What is the package's quality and reliability? If the package is plagued with numerous patches between scheduled releases, testing time for the larger system increases. Large systems tend to be long lived once they are fielded due to the significant investment in development. Therefore, the quality, reliability and trustworthiness of the COTS package are critical considerations. If the package includes features that are not going to be implemented in the larger system, can feature blocking be assured or will these functions be a potential cause for total system failure or degradation of service? Military systems are prime targets for hackers; many times hackers find their way into a government system by defects in COTS components. Another concern is obsolescence. Will the package become outdated by the time it is fielded in the larger system? This is a consideration because of the long development time of the larger system and because that such systems typically have a long half-life.

Evaluation of COTS Components. Once a set of potential packages are identified, the next step is evaluation of the contenders. The product evaluation should include quality and reliability as well as functionality and performance. In addition to the "black-box" evaluation of the product, the requirement for the product's functionality and its interface within the total system should be carefully defined and documented. The evaluation process should down-select the candidate packages to a small number. In some cases, our initial search for products located up to 100 potential packages and exhaustive investigation would reduce the number to a short list of 10 or fewer. More detailed product tests would then be performed on those packages.

Supplier Identification and Evaluation Process. Once the short list of products have been evaluated and some potentially eliminated from consideration, it is time for an assessment of the vendors of those remaining products. The initial phase of the supplier evaluation process should examine the package's overall score in the product evaluation, which include not only performance and reliability but other factors such as cost and other consumer evaluations. We typically would determine the top three contenders and proceed to the second phase with this set. This next phase would consider three factors: (i) the domain expertise of the vendor, (ii) the business and financial health of the vendor, and (iii) the results of an on-site process assessment of the vendor.

We felt that the selection of a COTS software supplier should be a variation of the selection process for any software supplier [12]. So the primary, but not sole, factor is the technical expertise of the vendor. In addition to technical concerns, there are business issues to consider. Is the supplier's company financially sound? Even if the source code is held in escrow, it may not be easily supported if the vendor has gone out of business. And even if the vendor stays in business, will the software continue to be supported? Customer service and responsiveness are factors to consider. Thus, we felt that an on-site visit would be required. Each of the business/financial analysis results and the supplier process assessment results served as GO/NO GO decision gates.

Software Supplier Assessment. The evaluation of a vendor should include not only the quality and reliability of the product but also the quality of the vendor's configuration management and release processes. These assessments do not need to be long, arduous evaluations. They can be streamlined to fit the size and scope of both the product and the vendor. Basically, the question is: What is the real cost of the software package? The total costs include not only the licensing, the integration and interface testing in the larger system, but also training, long-term maintenance, and the management of upgrades over time. The answers to these cannot be directly calculated but can be indirectly approximated by a combination of product evaluation, business analysis, and supplier assessment.

Two entities can greatly assist in the identification and evaluation of COTS components: the requirements specifications and the interface control document (ICD). The former helps to answer questions related to the applicability of the package; the latter helps to scope the level of effort needed to incorporate the package, thereby determining some of the hidden costs. The ICD is also a critical factor in testing of a large system with integrated COTS components. With a well-written and structured ICD, re-usable interface tests can be developed. The ability to reuse and/or automate tests becomes crucial. Incremental test planning is a necessity in any large-scale system because the system is evolving. Whenever COTS packages are incorporated, additional tests concerning error handling at the interface should be designed. In addition, performance and stress testing of the component's interface should be conducted. The buy versus make decision should be based on a realistic estimate of the total cost of each effort.

Structure of On-Site Visit. Because of the magnitude of software being contracted in the first system mentioned, we developed a supplier process assessment that could be tailored for the size of the company and for the type of software activity. If the potential supplier would be developing software, the visit ranged from one to three days depending on the size of organization. For COTS products, the visit was streamlined to a one-day visit regardless of the size of the company.

We structured the supplier process assessment using the SEI Capability Maturity Model as the basis. The vendor evaluation had many similarities to an SEI Software Process Assessment (SPA). Both use an interview and discussion format and we selected many of our questions from the SEI questionnaire. In addition, the lead of the evaluation team was always a trained SEI assessor. The evaluation teams generally were had a total of three members, which is a reduction from the SEI assessment team size. The development team which would be incorporating the package was always represented on the assessment team. The third member would come from any part of the development team or from the contracts organization which had professionals who specialized in software contracting. If these non-lead members were not certified SEI assessors, then they received a short in-house training course in the supplier evaluation process prior to visiting the vendor. One of the differences from the initial SEI assessment format is that we scheduled private interview sessions with a senior executive manager and with the chief technical officer or chief scientist or chief software engineer (depending on the vendor's organization). For the COTS vendors, we also spent more time with the Quality Assurance team and with the configuration management team than with the development and test teams. We also met with customer service and support teams of the potential COTS supplier. Another difference is that the SEI questionnaire, while a basis for the discussions, was not completed by the vendor in advance. Lastly, as with an SEI assessment, we did present a findings session at the close of the visit. The findings presentation included a supplier rating, but did not determine an SEI maturity level from 1 through 5. Rather, the rating was one of fully qualified, qualified, or not qualified. Most of the suppliers ranked in that middle category. However, in the most important portion of the findings, we listed what we perceived to be the strengths and weaknesses of the supplier. Improvement in weak areas became contract requirements if we chose to pursue the relationship further.

In a commercial satellite communications example, several packages related to orbital analysis and telemetry tracking and control that were efficiently incorporated due in large part to the careful supplier evaluation. In another case, the board of directors of a small company mandated the company's president to address the action items resulting from our supplier evaluation regardless of whether we entered into a contract. On the flip side, we terminated a contract with one supplier who had excellent domain expertise because the first deliverables from the organization were extremely poor; the root causes of the poor quality were the very areas identified as opportunities to improve from the supplier evaluation.

Assessment Follow-On. We instituted a mechanism of communications and follow-on with our contracted suppliers. This included a management forum of quarterly meetings of senior executives, primarily from those suppliers who were developing software. There was also a periodic technical forum that included all of our suppliers. This was especially important in the use of one COTS product, namely the mandated configuration management tool. We kept open the option of re-evaluation. These re-evaluations were based on contractually required improvements (if any), and issues of concern from the sub-contract managers. The follow-on visits were informal with mutual presentations, questions, and discussions. With all of the suppliers, we provided an opportunity to evaluate us and let us know how well we were doing as contract managers. We felt that this is an important aspect of growing a long-term supplier relationship. Total improvements can best be made in a spirit of constructive and honest feedback.

Conclusions and Lessons Learned. The following are some of the findings related to implementing supplier process and product evaluations in large-scale development, which include a mix of both commercial and government systems.

First, the use of COTS packages can reduce total system development time, but the savings are partially offset by increased design time up front and increased interface testing downstream. It is critical to scope out the total life-cycle cost of the COTS package, not merely the licensing cost. A formal COTS identification and evaluation process can greatly assist in this scoping effort. The COTS evaluation should include an assessment of the vendor's process as well as of the package itself. The decision to incorporate the COTS component should be based on product, process, and business factors.

Lastly, a hard lesson learned concerns whether or not to modify a COTS package. Since the package is most likely not a perfect fit, there is a tendency to work with the vendor and modify the package. The short version of the lesson is: DON'T. If you feel that you must modify the package, modify your process first. Then, and only then, if you absolutely must modify the package, build a wrapper and still do not modify the package. If that still doesn't convince you, perhaps simple economics will. If you modify the COTS package, you may void the warranty. If you contract with the vendor to modify the package, you will be contracting with them for the life of your product.

In summary, large-scale systems are difficult project development activities with significant time constraints and with anticipated but unknown changes. Inclusion of COTS components can reduce total life cycle costs and support the successful fielding of a quality product. The quality and reliability of that system can be improved with a COTS product and process evaluation.

References.

1. Tom DeMarco and Ann Miller, "Managing Large Software Projects", *IEEE Software*, July 1996.
2. Frederick P. Brooks, Jr., *The Mythical Man Month, Essays on Software Engineering*, Addison-Wesley, 1975.
3. Capers Jones, *Patterns of Software Systems Failure and Success*, International Thomson Computer Press, 1996.

4. Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice*, Addison-Wesley, 1998.
5. Daniel M. Marks, *Testing Very Big Systems*, McGraw-Hill, 1992.
6. W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques", originally published in *Proceedings of WESCON*, August 1970; also available in *Proceedings of 9th International Conference on Software Engineering (ICSE 9)*, IEEE/ACM, 1987.
7. Bill Curtis, Herb Krasner, Vincent Shen, and Neil Iscoe, "On Building Software Process Models Under the Lamppost", *Proceedings of 9th International Conference on Software Engineering (ICSE 9)*, IEEE/ACM, 1987.
8. Barry Boehm, "Anchoring the Software Process", *IEEE Software*, July 1996.
9. Ann Miller, "Design and Test of Large-Scale Systems", *Joint Proceedings of the International Conference on Software Management and International Conference on Applications of Software Measurement*, March 2000.
10. Watts Humphrey et al, *A Method for Assessing the Software Engineering Capability of Contractors*, Technical Report CMU/SEI-87-TR23, Software Engineering Institute, 1987.
11. IEEE Standard-1062, *Recommended practice for Software Acquisition*, Institute of Electrical and Electronics Engineers, 1993.
12. Jim Nielsen and Ann Miller, "Selecting Software Suppliers", *IEEE Software*, July 1996.

Maintaining COTS-Based Systems

Dr. Mark R. Vigder
John Dean
National Research Council of Canada
Institute for Information Technology
Ottawa, Ontario
Canada
K1A 0R6
{mark.vigder|john.dean}@nrc.ca

Summary: After deployment, all software systems require an extensive and expensive phase of maintenance and management regardless of whether they are COTS-based or custom built. Understanding how COTS-based systems are maintained, and why they are different from custom-built systems, can lead to systems that are better and more cost-effective over their lifetime.

1 Introduction

After deployment software systems enter a phase of maintenance, management, and evolution that can last many years until final decommissioning [3,5]. This post-deployment phase is the longest and hence the most expensive phase of the software lifecycle. Success during this phase is often the determining factor as to whether a software system is cost-effective over its lifetime.

Building a software system from COTS products does not change the importance nor the expense associated with maintenance, evolution and management. COTS-based systems must continue to satisfy evolving user requirements, failures of the system must be dealt with, the system must adapt to the ever-changing environment, and managers must be able to monitor and control the deployed system. These activities are necessary whether a system is built from scratch or built using commercial products.

The nature of the post-deployment activities changes when dealing with COTS-based systems rather than with custom built systems. If COTS-based systems are to be successful over the many years that they are expected to be in service, organizations involved in building or acquiring COTS-based systems must understand and accommodate these differences.

2 COTS-based systems: why is maintenance different?

Software maintenance includes all the activities required to evolve a software system over its lifetime. Although the motivation for maintaining COTS based and custom systems is the same, the nature of the activities required of the maintenance personnel is different. The different activities required for maintaining COTS intensive systems arise for a number of reasons.

Primary among the reasons for the different maintenance activities is the fact that the evolution and upgrades for the individual COTS products are outside the direct control of the system developers and acquisition organizations. The COTS products are maintained and supported by the COTS product developer or their agent. The system developer must treat these products as single black-box entities with little or no visibility into the internals of the product and perform the maintenance at the level of large-scale products rather than at the source code level. The only source code being maintained by the system developer is that required for integrating the large-scale COTS products. This includes code for wrapping and tailoring the individual products, as well as the "glue code" required to connect the products together. Wrapping and tailoring of

NRC Report 43626. This work was funded by Defence Research and Development Branch, Dept. of National Defence, Canada.

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

the products (without accessing the products source code) becomes necessary to overcome architectural mismatch between products, to customize the product to conform to local requirements, and to build workarounds to overcome the inevitable bugs (and features) that are included in any COTS product.

From the acquisition agencies' perspective, they have effectively ceded control over maintenance and evolution of large parts of the system to outside commercial agencies. Maintenance of the COTS intensive system is now driven in a large part by the vendors of the different products rather than by the system developer. In effect, having amortized the cost of development and maintenance among many different users, acquisition agencies are now one among many users driving the direction of the COTS software evolution.

2.1 Maintaining a COTS-based system

In order to more effectively maintain and manage COTS-based systems it is necessary to identify the activities of the maintenance and management personnel. Once the activities have been identified, strategies can be developed to facilitate these activities. COTS-based maintenance and management, although similar in many respects to maintaining custom-built systems, has qualitative differences. These differences result in the following activities in the post-deployment phase (Table 1).

Component reconfiguration. Reconfiguring components is the act of replacing, adding and deleting components within the system. Reconfiguration occurs for many reasons, perhaps the most common being the frequency with which commercial product vendors release updated versions of their software. It is not uncommon for each product to be upgraded two or three times per year. Often, system integrators are forced to replace older product versions with the upgrades in order to fix bugs or improve functionality. Other reasons for reconfiguring the components are to replace aging components with better products from competing vendors, or to add and delete products as the functional requirements of the system evolves.

Reconfiguring the components is an expensive activity requiring the integrators to go through a complete release cycle including product evaluation, testing, design, integration, and system regression testing.

Troubleshooting and repair. All systems fail and COTS-based systems are no different in this respect. However, with COTS-based systems maintenance and management personnel generally cannot look inside components when trying to isolate the cause of the failure. Information must be gathered by experimenting at the edges of the components. Identifying the source of the fault requires running a series of experiments to determine the product or products causing the problem [2].

Identifying and fixing the fault is no longer an activity performed solely by the system builders. Having used third-party products, system builders must now work closely with the support staff of the product suppliers, and with the general product user community. Where faults involve complex interactions involving sets of products from different vendors, many different organizations may be involved in the troubleshooting and repair of the system.

Configuration management. For COTS-based systems configuration management is done at the level of products rather than at the level of source code. Issues that maintainers must address include: change history for each individual product; availability and support level provided by the product vendor; management of configurations of the COTS-based system that are installed at each deployed site; compatibility requirements and constraints between sets of products; and licensing issues associated with each product.

Maintenance activity	Description
Component reconfiguration	Updating product versions, replacing COTS products with similar products, adding/deleting products
Troubleshooting	Identifying causes of failures among sets of COTS products, developing workarounds with the products, liasing with the COTS product maintainers
Configuration management	Tracking versions of different COTS products, tracking deployment configurations, determining compatible versions of products
Testing and evaluation	Testing new product versions as they become available, within the context of the system into which they will be integrated
Tailoring user level services	Enhancing the services available to the end user by configuring COTS products, combining services of multiple products, etc.
System monitoring	Monitoring different aspects of system behaviour, such as communication, resource usage, process invocation, etc.

Table 1. Maintenance/management activities for COTS-based systems.

Testing and evaluation. Testing and evaluating COTS products is an ongoing activity during maintenance. New product versions as well as new products must be evaluated for inclusion within the system and products must be tested during operational use.

Tailoring user level services. COTS products provide a generic functionality that can be used by many applications and organizations. System integrators must customize and tailor this functionality to satisfy the local operational requirements that are unique to the end-user organization. Successful systems are those that can be quickly modified and tailored to meet evolving user requirements.

For COTS-based systems tailoring involves an ongoing process of customizing and configuring products, adding new components to the system, and combining services of multiple products in novel ways. Since integrators do not have access to product source code this must be done through gluing products together to provide enhanced functionality and using vendor supported tailoring techniques to customize the products.

System monitoring. System managers and maintainers must continuously monitor a system during its ongoing operation. This must be done to measure performance and resource usage, watch for failures, and determine user behavior. Because COTS software is black box, with limited visibility into internal behavior, monitoring for maintenance purposes can be difficult to do effectively.

3 Planning for post-deployment

Systems are maintainable and evolvable through their lifetimes because they were explicitly designed to be so. Maintainability cannot be built in "after the fact" but must be considered during the early stages of analysis and design.

For COTS-based systems, there are two phases of construction during which system builders must consider maintainability and evolvability. The first is during product evaluation and selection. The products used to build the system have a great deal of impact on the characteristics of the system during its maintenance.

The second phase of construction that impacts system maintenance and evolution is the architecture and design of the system. Different architectural styles have different properties relative to the evolvability and maintainability of a system. By identifying the properties required of a COTS-based system an appropriate architectural style can be applied that provides these properties.

3.1 Product selection

System builders do not control the individual products, but they do control which products are selected for integration into the system. There are many different criteria used for product selection but system evolution should be one of the factors considered when developing criteria for product selection. A number of properties of a product affect the long-term evolution and maintenance of the system.

Openness of the component. A component is open if it is designed to be visible, extendible and easily integrated into a wide array of systems. In general, the more open a component the easier it will be for maintainers and managers to monitor, manage, extend, replace, test, and integrate. Many factors combine to make a component open and among things to consider are: adherence to standards; availability of source code perhaps through open source licensing; and ability to interwork with products from many different vendors.

Tailorability of the product. Much of the maintenance effort for COTS-based systems involves tailoring the functionality to meet evolving user requirements. One of the criteria for product selection should be the ease with which the product can be tailored to satisfy local requirements. Although products are black box and the source code cannot be modified there are many techniques product builders can use to make their products tailorable. Examples of tailoring techniques include scripting interfaces, data configuration files, and frameworks that can be extended through the use of inheritance and plug-ins.

Available support community. System builders require extensive assistance from external organizations to support commercial software. This support comes from the vendor and the user community. Given that successful maintenance is dependent on this support, system builders must evaluate the support available for the product during the product evaluation process.

3.2 Designing for evolution

System builders do not own COTS software, but they do own the architecture and design used to integrate the software. By addressing issues of maintainability during the software design activity, designers can build a system that facilitates the maintenance activities associated with COTS-based systems and avoids many of the pitfalls [1].

There are two major issues that can be addressed when designing COTS-based systems for maintainability. The first is the management of dependencies between the diverse software elements. Many uncontrolled dependencies between products make it exceedingly difficult to modify or analyze a software system. Component replacement or addition will be difficult due to the affects that can ripple through the various component dependencies. Many interdependencies also make understanding failures and isolating faults a more complex task.

The second design issue that must be addressed is visibility into the system. A system is visible if maintenance and management personnel can instrument and monitor the system. This involves querying the system to determine its operational characteristics, current configuration, fault incidents, etc. Visibility is a necessary characteristic for testing and managing systems. For COTS-based systems, where there may be limited visibility into the individual products, designers must build visibility into the architecture.

3.2.1 Managing product dependencies

Complex and intricate product dependencies result in a fragile system in which it is difficult to upgrade, replace, add and remove components. To alleviate these problems,

Architectural view	Entities
Interconnection topology	Map of the data flow between the COTS components.
Connection infrastructure	Mechanism used to transfer data and control among the software elements, e.g., CORBA, DCOM, RMI.
Interfaces	Exposed parts of the COTS software product.
Collaborations	Ongoing behaviour required among a set of components in order to provide a service.
Environment	Dependencies made by the COTS product about the environment in which they are operating, e.g., operating system, software versions, file structure, etc.
Control mechanism	Dependencies caused by assumptions about process structure, control flow, activation, etc.

Table 2. Causes of component dependencies.

designers must manage the dependencies between the products so that COTS-base maintenance is possible.

There are many ways that software components within a system may be dependent. Some of these are explicit, such as the direct transfer of data through an exposed interface. Other dependencies are implicit and subtle, such as conflicting assumptions that different software products can make regarding the environment under which they are executing.

Table 2 summarizes the major causes of component dependencies. It is important for designers to recognize that they cannot eliminate dependencies, but they can manage them in a way that allows for more effective maintenance [4].

3.2.2 Designing for visibility

Visibility is a property of a system that permits inspection and instrumentation by managers and maintainers. Capabilities involved include event logging, raising alarms, determining the current configuration, etc. Visibility is necessary for debugging, testing, isolating faults and managing the system.

Designers have little or no control over the visibility provided by the individual software products. However, through the architecture and design a great deal of visibility can be built into the system by using the glue and integration code as tools for monitoring and viewing the system. An example is shown in Figure 4 in which a mediator exposes a management interface. Through this interface different information about the collaboration and the components can be gathered such as the events generated and received by the components, activiations of the components, component versions, etc.

4 Conclusions

Although component-based software systems provide many advantages, designers and users must still expect that the majority of the lifecycle cost will be incurred after the initial deployment of the system. Reducing this cost, and easing the maintenance and management effort, requires designers and architects to consider the post-deployment activities during the earliest stages of software development. By identifying the activities that maintenance and management personnel perform to support component-based systems, and using a design that supports these activities, systems can be made more cost-effective.

Bibliography

[1] David Garlan and Robert Allen and John Ockerbloom. Architectural

Mismatch or Why it's hard to build systems out of existing parts. In 17th International Conference on Software Engineering, pp179-185 1995.

[2] Scott Hissam. Correcting System Failure in a COTS Information System. In Proceedings of the International Conference on Software Maintenance, pp170-176, Nov 1998.

[3] Duane W. Hybertson and Anh D. Ta and William M. Thomas. Maintenance of COTS-Intensive Software Systems. Journal of Software Maintenance, 9(4):203-216, 1997.

[4] Mark Vigder and John Dean. Building Maintainable COTS-Based Systems. In International Conference on Software Maintenance, pp132-138, 1998.

[4] Mark Vigder. The maintenance, management, and evolution of component-based systems. In *Component-Based Software Engineering: putting the pieces together*. Adison-Wesley, to be published, 2000.

[5] Jeffrey Voas. Disposable Information Systems: The Future of Software Maintenance?. Journal of Software Maintenance: Research and Practice, 11:143-150, 1999.

Detection of Malicious Code in COTS Software via Certifying Compilers

Robert Charpentier and Martin Salois
{Robert.Charpentier@drev.dnd.ca, Martin.Salois@drev.dnd.ca}

Defence Research Establishment Valcartier
2459 Pie XI Blvd. North
Val Bélair, Québec, Canada
G3J 1X5

April 2000

Abstract

Information technology is more and more a vitally important underpinning to our economy and to our society. It is embedded in everyday applications and animates a wide class of systems that range from small to large and from simple to extremely sophisticated. Among the probable threats for military information systems, the presence of malicious code within COTS applications has been identified as a major risk that has not received a lot of attention. Like a virus that has infiltrated an information system during an electronic information exchange, malicious code integrated into a commercial application could remain undetected and present a major risk for the safety of information within a military system. In this paper, techniques to detect malicious code within commercial applications are reviewed. Emphasis is placed upon the certifying compiler, which enforces a formal security specification while compiling the source code. This emerging technology offers the most comprehensive and sustainable approach for large applications and for the periodic certification of upgrades.

1 Introduction

The Defence Research Establishment, Valcartier (DREV) carries out an extensive R&D program in Command and Control Information Systems (CCIS) for the Canadian Department of National Defence (DND). During the Information Warfare Workshop held in Ottawa in Oct. '96, several R&D challenges were identified and presented to DND and industry representatives [17]. Trusted software design and validation was one of the areas where additional effort was deemed necessary to meet DND needs. Of particular concern was the integration of Commercial-Off-The-Shelf (COTS) software into military information systems.

Exploiting COTS software through integration poses a distinct dilemma. On one hand, COTS software is very attractive; its use promises to reduce development time and costs. On the other hand, it introduces new risks into military information systems: hidden functionalities, trap doors, private control codes giving enhanced privileges, logical or temporal bombs [6], etc.

A feasibility study completed in 1998 indicates that a variety of software analysis techniques can be applied to the management of the risk associated with COTS software in military information systems. Among them, the exploitation of certifying compilers appears to be a very powerful technology for the efficient yet exhaustive verification of software with minimal human supervision. This paper summarises the lessons learned in the MaliCOTS project, carried out jointly by DREV and Laval University. The proposed strategy will, after successful implementation, ensure the safe integration of previously untrusted software in military information systems via certifying compilers.

2 Malicious Code

Malicious codes are fragments of programs that can affect the confidentiality, the integrity, the data and control flow, and the functionality of a system without the explicit knowledge and consent of the user. We distinguish between intentionally malicious and unintentionally malicious code. Malicious individuals who, for example, use such programs to access confidential data generally introduce the first. The second is due to inadvertent human error, especially during development of the software.

To detect malicious code in COTS software, one must be able to distinguish between its types. Starting from the taxonomy proposed by McDermott & Choi [12], a new taxonomy has been defined that is specifically intended to facilitate the detection of malicious code in COTS soft-

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

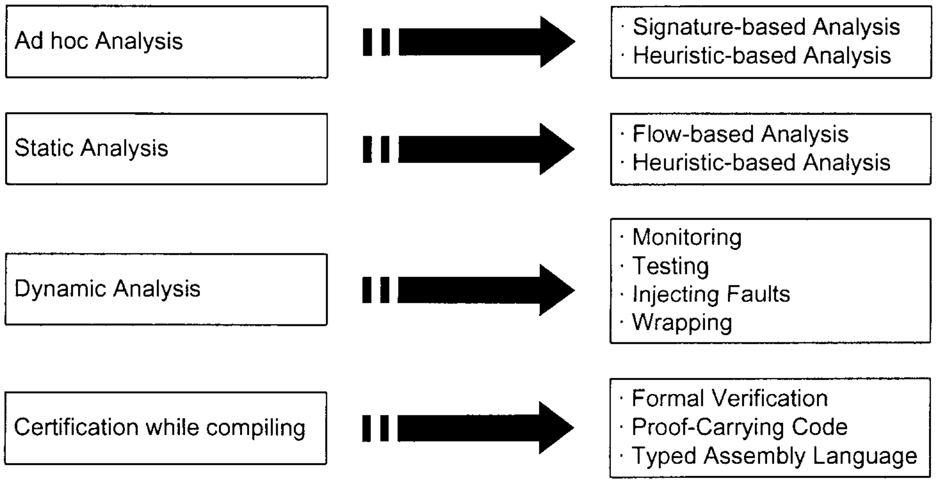


Figure 1: Potential techniques to detect malicious code in COTS software

ware [3].

One of the key concepts of the MaliCOTS project is always to refer to a security policy to distinguish an acceptable activity from a potential threat. It reflects the fact that software functionalities can never be considered malicious in and of themselves; even reformatting a disk or destroying a file are useful operations in certain circumstances; that is why such capabilities were devised and made available to system users. But in many operational contexts these functions should not be made available to end-users because of the associated risks. The most rigorous way to enforce such a policy is to formalise these constraints explicitly in a security specification based upon permissible access mechanisms. This strategy is documented in more detail in this paper.

In practice, threat that system analysts are typically concerned with are:

- the presence of trapdoors in COTS packages (as found in Unix, Windows NT & '98 [6, 9]),
- license expiration logic [16],
- hidden communications (e.g., a CD player software that is reported to send 'your listening preferences' to a distributor periodically [15]), and
- other undesirable functionalities such as those present in the flight simulator in Excel '97 and the Word '97 pinball machine [1, 2].

The next section summarises the feasibility study, completed in fall 1998, into ways to detect such malicious code.

3 Technology Options to Detect Malicious Code in COTS Software

Figure 1 identifies a variety of techniques applicable to the MaliCOTS project, in order of increasing level of complexity.

Reference [4] contains a comparative analysis of these techniques. In summary, ad hoc techniques consist of code inspection in search of a known malicious signature or its generalisation (often called a heuristic). This approach has been very successful in detecting viruses within exchanged files, but its effectiveness in detecting malicious code in large software applications is limited since a priori knowledge is needed (i.e., either signature or behaviour profiles).

Static analysis of code comes from the world of program optimisation and software analysis. It consists of examining the code (perhaps in some abstract representation) without running it. At present, static analysis is essential to COTS certification because it gives a relatively precise idea of program behaviour for all possible execution conditions. However, the technique is limited in capability, especially when source code is not available, which is typically the case for COTS. The process requires enormous human effort for very large applications [5].

Dynamic techniques examine the behaviour of the code while it is running. Such analysis is a pragmatic approach that offers short-term benefits. Many variants are available: monitoring execution, running an exhaustive suite of tests, injecting faults in critical variables or wrapping the commercial code into a software shell that detects and filters out unwanted activities. Another paper presented in this conference deals with dynamic detection and provides supplementary information [18].

Each of these techniques has its place and offers short-term solutions to the detection of malicious code in COTS software. However, they are all reactive, in the sense that they evaluate the COTS package after development, when detection is made more difficult by the lack of access to source code.

Being unsatisfied with this situation, we have searched for a truly innovative approach to COTS integration that will overcome existing difficulties with the non-availability of source code, with time-consuming manual

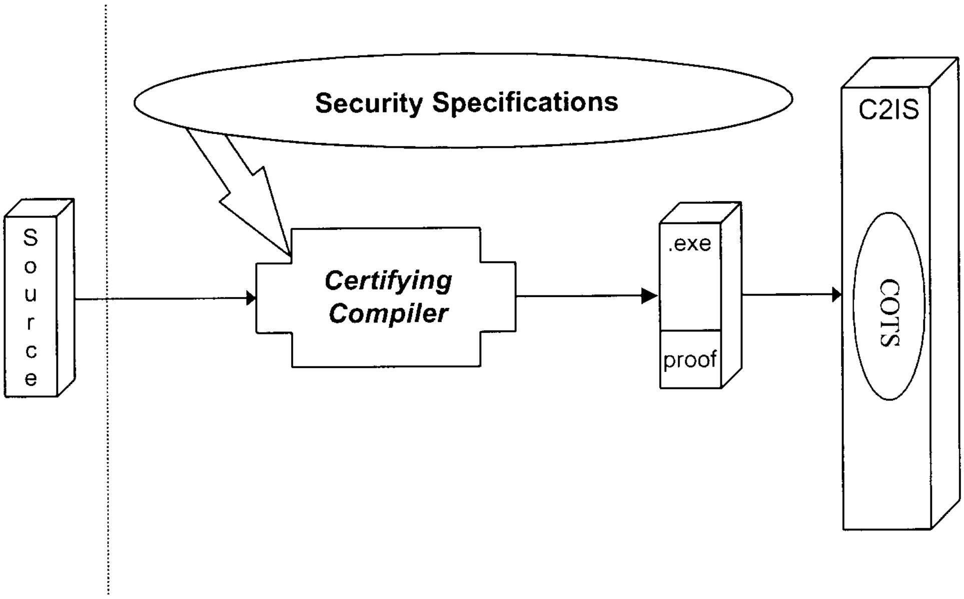


Figure 2: Certifying compiler — basic concept

inspection of software and with difficulty in ensuring the completeness of verification.

Certifying compilers have emerged as an extremely powerful technology to manage the risk associated with COTS integration. The basic idea is to put enough "intelligence" into the compiler that it will not only produce the executable code but also perform formal security verification. As shown in Figure 2, the compiler needs two inputs: the source code and the security policy. The compiler then translates the source code into the appropriate intermediate language (e.g., assembly, byte code, etc.) along with embedded security annotations.

The next section describes the concept and gives a practical feel of its capabilities in our particular context.

4 Certifying Compilers; Concept & Practice

4.1 Concept

As in human health, prevention is certainly the best cure. So it is worthwhile from a security standpoint to elaborate methodologies that guarantee that COTS software products are free of any malicious code from the start. In order to do that, we propose the inclusion of intelligence in the compiler to allow enforcement of a security specification while compiling.

Figure 3 illustrates the most general scheme to produce trusted software while compiling. The first step consists of compiling the source code and introducing static annotations in the object file (i.e., byte code for JAVA, assembly language or other intermediate language). It is a rather simple and mechanical process to introduce the annotations. Secondly, the annotated code is submitted to a verifier (or a verifying linker) that enforces a formally expressed security specification. By doing so, the final executable application can be assembled safely and sealed with a security tag before integration into a critical information system.

This is a very flexible approach. Not only can the annotations be produced rapidly and independently of the final integration but also different local security policies can be enforced in different parts of an organisation on a single annotated component. Another great advantage of this approach is that there is no need for the software integrator to have access to the source code. The only requirement for the software producer is to adopt an annotation structure that the integrator can recognise and verify for correctness. This key feature protects the intellectual property of software producers.

The second step of the process (verification) starts with a comparison of the annotations with the object code. Any anomalies in the compliance of the code with the annotations can easily be flagged for further investigation. In other words, if the code is modified after it was annotated, or if the annotations are changed without any code modification, the verifier will rapidly detect it. The only component that one must trust in this system is the verifier itself; there is no need for trustworthiness in the code producer, the annotating compiler or the transmission channel up to the verifier. This is a very important feature for security architects, who may deal with the trustworthiness of only one component, the verifier.

4.2 Annotation Structures

So far we have not described the content and the structure of the annotations the compiler produces. Many options exist, each with its advantages and disadvantages. In

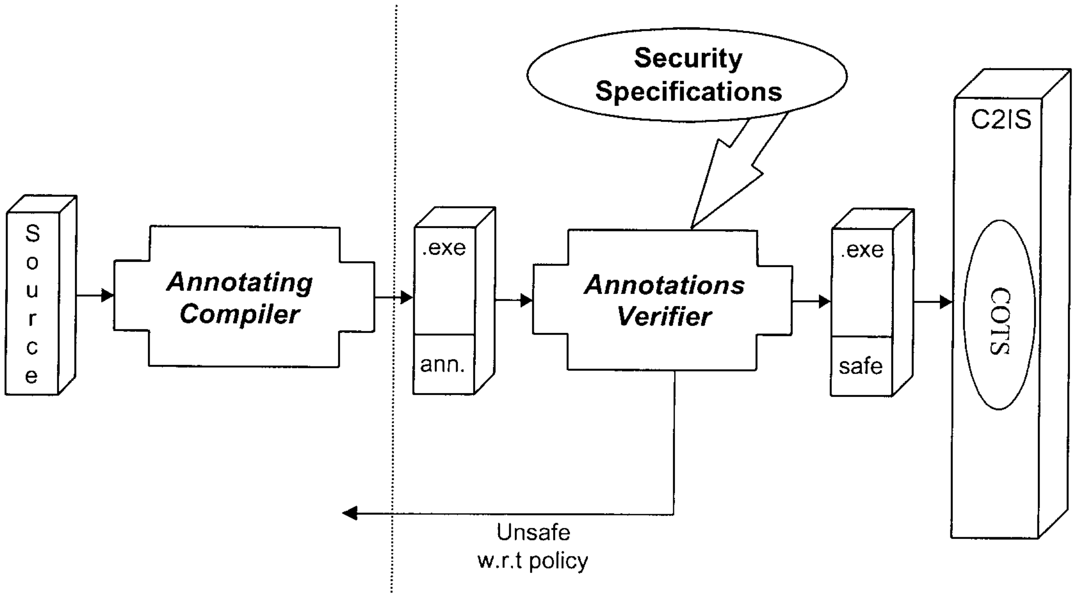


Figure 3: Certifying compiler — generalized concept

the MaliCOTS projects, we examined three possibilities closely:

- PCC (Proof Carrying Code), developed under the leadership of Peter Lee and George Necula at CMU (Carnegie Mellon) and at Berkeley University;
- ECC (Efficient Code Certification), led by Dexter Kozen from Cornell; and
- TAL (Typed Assembly Language), designed by Greg Morissett at Cornell University.

PCC is a technique to ensure the safe execution of untrusted mobile code. When code is transferred between a client and a producer, the producer must append to the code a formal proof that it is consistent with some shared security policy. The client can easily check the proof by using a simple and easy-to-trust proof checker. PCC is a very comprehensive and secure approach [14].

ECC was designed to be a much lighter solution to code certification. The annotations contain structured information that qualifies the safety of the code. It was designed for efficiency and performance, sacrificing some of the rigour of other approaches [10].

TAL proposes to introduce "type"-typing information-into the code. Basically, software types are static descriptors of logical entities (e.g., variables, constants, character strings...) and of how they are used in the code. These annotations are light and informative and can easily be produced and managed within a comprehensive security policy [13].

TAL was selected as the technology of choice for the detection of malicious code in COTS software. Type annotations provide an automatic way to verify that a program will not violate safety properties and, potentially, high-level security requirements. At this time, TAL can handle:

- control flow safety (i.e., programs cannot jump to code that was not verified and stack preservation is enforced),
- memory safety (i.e., access to initialised memory locations and array bounds checking) and
- type safety (i.e., the compatibility of types in operations).

Complementary annotations in the "ECC style" will be considered later in the MaliCOTS project if they are needed.

In summary, type annotations are static approximations of the behaviour of the program. Essentially, they correspond to typing preconditions on code labels. Before transferring control to any label, the register, stack and relevant variables must contain values of the types specified. The type-checker matches each instruction operand against these constraints to ensure that they do not violate safety properties.

4.3 Example

To illustrate the concept of annotations, we will now examine a simple program written in C (Code Excerpt 1) and compile it to assembly language with annotations (TALx86 code) as shown in Code Excerpt 2, where annotations appear in bold. An expression such as "eax: B4" indicates that the register "eax" must contain four bytes if the following instruction is to be executed. Inference rules are used to verify formally that all conditions are met before the activation of a given operand (e.g., an arithmetic operation or a call procedure like those shown in Figure 4).

Code Excerpt 1: Sample C code

```
#define TABLEAU 100
unsigned int premiersTABLEAU;

int estPremier( int nombre, int compte )
{
   int i = 0;
   int iPremier = 1; //sans preuve du contraire, c'est un nombre premier

   for (i = 0; i < compte; i++)
   {
      if ( nombre % premiersi == 0 )
      {
         iPremier = 0;
         break;
      }
   }

   return iPremier;
}
```

$$(ArithBin) \qquad \frac{\varepsilon \vdash op_1 : B4 \qquad \varepsilon \vdash op_2 : B4 \qquad \varepsilon \vdash ValidBinops(op_1, op_2) \qquad \varepsilon \vdash Writeable(op_1)}{\varepsilon \vdash arithbin\ op_1, op_2 : \varepsilon}$$

$$(Call) \qquad \frac{\varepsilon \vdash cop : \{g_1\} \qquad \varepsilon \vdash g_1(\mathtt{esp}) = sptr\{g_2 :: c'\} \qquad \varepsilon \vdash \varepsilon.\gamma[\mathtt{esp} : sptr\{g_2 :: (\varepsilon.\gamma(\mathtt{esp}))\}] \preceq g_1}{\varepsilon \vdash \mathtt{call}\ cop : \varepsilon[\gamma : g_2]}$$

Figure 4: Two inference rules enforcing annotation checking in TAL

As part of the MaliCOTS project, we are developing an ANSI C compiler that will produce assembly language for x86 processors along with the corresponding TAL annotations. Our compiler is based on LCC (Lean retargetable C Compiler); a public-domain compiler that is well documented and for which source code is available [7]. A beta version of our TalCC compiler is available for government release, to allow a broader community to become familiar with the annotation technology. More information can be obtained from the authors of this paper.

For next year, we are planning the development of a JAVA annotating compiler that exploits the same annotation structure as TalCC. It will probably be based on JIKES, an IBM shareware compiler that is part of Linux packages. Emerging commercial products will also be considered ([11]).

5 Discussion & Conclusion

In view of budget reductions and decreasing human resources, integration of COTS software appears to be the only sustainable approach for Canadian DND [8]. At the present time, system analysts have only such labour-intensive techniques as static and dynamic verification to certify COTS software. It is expected that these techniques will remain useful (and mandatory, in many instances) for the certification of COTS packages. The MaliCOTS team values them greatly and attempts to integrate them into a common framework.

However, it is evident that more efficient and less time-consuming techniques are needed to handle COTS software, especially when periodic upgrades must be certified and when security policies must be met that vary significantly throughout an organisation.

Certifying compiler is a powerful enabling technology to meet this challenge. By formally specifying local security policies and by annotating an intermediate form of the code, the whole process is brought under control. Marginally acceptable functionalities and suspicious code segments may require later manual inspection, but the software core can be certified autonomously by the verifier.

This approach is also general enough to contribute to other kinds of certification, including interoperability compliance, reuse policy, maintainability specifications, etc., which are not examined by the MaliCOTS team at this time. Once these additional policies are expressed formally, simply passing the verifier over the annotated code would enforce them. Even though they are simple and compact, type annotations are very expressive. Our R&D on the detection of malicious code confirm that they have a strong potential for structuring and normalising the integration of COTS software into critical systems.

The expected benefits of certifying compilation are extensive and far-reaching. We hope that this paper will create enough interest in the technology that international collaboration can be organised to explore this ambitious certification paradigm more fully.

References

[1] T. E. E. Archive. Excel 97 Flight to Credits. `http://www.eeggs.com/items/718.html`.

[2] T. E. E. Archive. Pinball in Word 97. `http://www.eeggs.com/items/763.html`.

[3] J. Bergeron, M. Debbabi, J. Desharnais, B. Ktari, M. Salois, and N. Tawbi. Skeleton of a Taxonomy for Malicious Code. Technical report, DREV, Nov. 1998.

[4] J. Bergeron, M. Debbabi, J. Desharnais, B. Ktari, M. Salois, N. Tawbi, R. Charpentier, and M. Patry. Detection of Malicious Code in COTS Software : a Short Survey. In *First International Software Assurance Certification Conference (ISACC'99)*, Washington D.C., Mar. 1999. Section C1.

[5] J. Bergeron, M. Debbabi, M. M. Erhioui, and B. Ktari. Static Analysis of Binary Code to Isolate Malicious Behaviors. In 4^{th} *International Workshop on Enterprise Security*, Stanford University, California, USA, June 1999. IEEE Computer Society Press.

[6] J. T. Egan. Information Security Threats to Software Development. In *Software Technology Conference*, USA, Apr. 1997.

[7] C. Fraser and D. Hanson. *A Retargetable C Compiler: Design and Implementation*. Addison-Wesley, 1995. ISBN 0-8053-1670-1, q. `http://www.cs.princeton.edu/software/lcc/`.

[8] C. M. Hanrahan. Changing the Culture (COTS vs. Development). In *COTS Software Seminar*, Ottawa, Feb. 1998.

[9] G. Hoglund. A *REAL* NT Rootkit, Patching the NT Kernel. *Phrack Magazine*, 9(55), Sept. 1999.

[10] D. Kozen. Efficient Code Certification (ECC). Technical Report TR98-1661, Cornell University, Jan. 1998.

[11] P. Lee and G. C. Necula. Cedilla systems inc. `http://www.cedillasystems.com/`.

[12] J. P. McDermott and W. S. Choi. Taxonomy of Computer Program Security Flaws. *ACM Computing Surveys*, 26(3):211–254, Sept. 1994.

[13] G. Morrisett, K. Crary, N. Glew, D. Grossman, R. Samuels, F. Smith, D. Walker, S. Weirich, and S. Zdancewic. TALx86: A Realistic Typed Assembly Language. In *ACM SIGPLAN Workshop on Compiler Support for System Software*, May 1999.

[14] G. C. Necula. Proof-Carrying Code. In *Proceedings of the 24th ACM Symposium on Principles of Programming Languages*, pages 1–14, Paris, France, Jan. 1997. `http://www.cs.cmu.edu/~necula/pop197.ps.gz`.

[15] NTSecurity.net. Leap of Faith Now Required for Real Networks? `http://www.ntsecurity.net/forums/2cents/news.asp?IDF=173&TB=news`, Nov. 1999.

[16] Quarterdeck. Aids Information Kit Trojan. `http://www.quarterdeck.com/quarc/00000/00000030.htm`, June 1994.

[17] R. Roy, editor. *Strategic Investment Workshop Proceedings*. CRAD, Oct. 1996. Canadian Eyes Only.

[18] M. Salois and R. Charpentier. Dynamic Detection of Malicious Code in COTS Sofware. In *Commercial Off-The-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"*, Neuilly-sur-Seine Cedex, France, Apr. 2000. NATO, RTO.

6 Acknowledgements

The authors wish to thank the MaliCOTS collaborators who contribute to this ambitious research effort. Currently, 12 Laval University graduate students and 4 professors are involved. Special thanks must be addressed to Dr Mourad Debbabi, who has led this team with competence and dedication.

Code Excerpt 2: Corresponding assembly language with annotations in TAL

```
_estPremier:
 LABELTYPE < All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] ({ ESP: sptr[S(0)] B4::B4::s1 ,
   EBP: sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 , EBP: sptr[S(n1)] s1 } >
push ebx
push esi
push edi
enter 8,0
L8:
 LABELTYPE < All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^u::B4^u::B4^x::B4^x::B4^x::B4^x::
   ({ ESP: sptr[S(0)] B4::B4::s1 , EBP: sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 , EBP:
   sptr[S(-8)] B4^u::B4^u::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(0)] B4::B4::s1 , EBP:
   sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 } >
mov dword ptr (-4)[ebp],0
mov dword ptr (-8)[ebp],1
mov dword ptr (-4)[ebp],0
jmp tapp( L5, < s1, n1 > )
L2:
 LABELTYPE < All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^rw::B4^rw::B4^x::B4^x::B4^x::B4^x::
   ({ ESP: sptr[S(0)] B4::B4::s1 , EBP: sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 , EBP:
   sptr[S(-8)] B4^rw::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(0)] B4::B4::s1 , EBP:
   sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 } >
mov edi,dword ptr (20)[ebp]
mov eax,edi
mov edi,dword ptr (-4)[ebp]
mov edi,dword ptr (_premiers)[edi*4]
xor edx,edx
div edi
cmp edx,0
jne tapp( L6, < s1, n1 > )
L9:
 LABELTYPE < All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^rw::B4^rw::B4^x::B4^x::B4^x::B4^x::
   ({ ESP: sptr[S(0)] B4::B4::s1 , EBP: sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 , EBP:
   sptr[S(-8)] B4^rw::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(0)] B4::B4::s1 , EBP:
   sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 } >
mov dword ptr (-8)[ebp],0
jmp tapp( L4, < s1, n1 > )
L6:
 LABELTYPE < All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^rw::B4^rw::B4^x::B4^x::B4^x::B4^x::
   ({ ESP: sptr[S(0)] B4::B4::s1 , EBP: sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 , EBP:
   sptr[S(-8)] B4^rw::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(0)] B4::B4::s1 , EBP:
   sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 } >
L3:
 LABELTYPE < All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^rw::B4^rw::B4^x::B4^x::B4^x::B4^x::
   ({ ESP: sptr[S(0)] B4::B4::s1 , EBP: sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 , EBP:
   sptr[S(-8)] B4^rw::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(0)] B4::B4::s1 , EBP:
   sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 } >
inc dword ptr (-4)[ebp]
L5:
 LABELTYPE < All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^rw::B4^rw::B4^x::B4^x::B4^x::B4^x::
   ({ ESP: sptr[S(0)] B4::B4::s1 , EBP: sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 , EBP:
   sptr[S(-8)] B4^rw::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(0)] B4::B4::s1 , EBP:
   sptr[S(n1)] s1 , EAX: B4 })^x::B4::B4::s1 } >
mov edi,dword ptr (24)[ebp]
cmp dword ptr (-4)[ebp],edi
jl tapp( L2, < s1, n1 > )
...
```

Application of COTS Communications Services for Command and Control of Military Forces

Peter Kerr[1] Jeff McCarthy[1,2]

[1]Head Wireless Systems Group, Communications Division, Defence Science and Technology Organisation, Australian Department of Defence, United Kingdom

[2]Satellite Communications Department, DERA Defford (on attachment), WORCS WR8 9DU, United Kingdom

Contact: Peter.Kerr@dsto.defence.gov.au
JMCCarthyl@dera.gov.uk

1. Introduction

This paper describes issues related to the use of commercial communication systems in support of military command and control. These systems[1] provide paging (messaging) and telephony services with global reach using small (personal), autonomously powered terminals.

New commercial telephony and paging systems offer ready access to advanced communications technology for a range of benign and hostile forces including the military, government agencies, media organisations, emergency services, insurgents and terrorists. The size, cost, coverage and ubiquity of all of these systems combined with the availability of tools targeting internet application development creates an interesting mix of threat and opportunity for military organisations.

One of the key advantages offered by the group of new telecommunications networks is diversity. Diversity of supply may enable a future adversary to use up to five systems in order to provide a voice service. For example, a user could subscribe to voice services based on GSM, CDMA, Inmarsat, Iridium, Globalstar systems using only three terminals that could easily fit into a briefcase. These example systems would operate in five different frequency bands and all are highly independent of each other in terms of the supporting network.

This paper is structured in the following way. Section two describes some high level attributes required of these commercial systems in order to operate in a military communication environment. Section three highlights the differences that would typically exist between the commercial and military communication markets and their associated procurement strategies. Section four provides some examples of COTS solutions for military applications, which include Command and Control Warfare (C2W), and the application of COTS for Australian Defence Force (ADF) communications.

2. Military communication environment

The current thrust in military communications is towards achieving C4ISR[2] dominance in the battlespace. This dominance will provide commanders with the situational awareness and understanding that is necessary to achieve decision superiority at the tactical and operational levels of warfare.

In order to achieve the aforementioned objectives an integrated communication system will be required to provide a high level of connectivity between the various sensors, weapon systems, and Command and Control (C2) elements that exist in the battlespace. This leads to the concept of Network Centric Warfare (NCW), in which the battlespace consists of a dense grid of sensor and shooter networks that have been seamlessly integrated through communications onto a common information grid as shown in Figure 1.

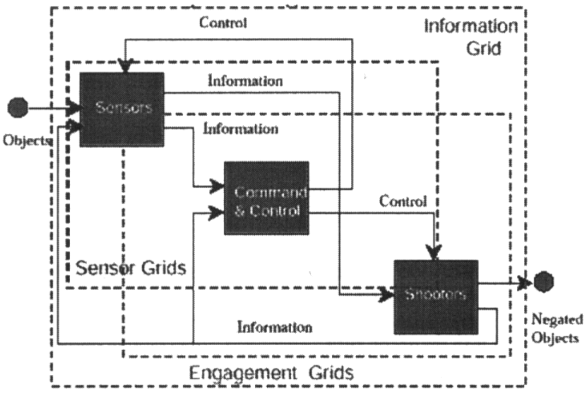


Figure 1: Integration of C2 in Network Centric Warfare

Current concepts for network centric warfare assume a communications architecture that is able to support a broad range of bandwidths over both short and long distances[3]. To a large extent this architecture is expected to depend on broadband wireless communications to support these information flows, however, the pre-eminent communication services required for command and still remain voice and low rate data.

It is the belief of the authors that the voice and low rate data command and control communication services could be provided by commercial satellite systems which, despite their commercial nature, may serve as a catalyst for future military communications and such concepts as NCW. The difficult problem for military planners, however, is tailoring these commercial systems in order to confer military attributes onto systems that have been designed for commercial operations. The first step towards solving this problem is, identifying the desired military attributes that are expected of information flows

[1] The COTS solution in this case may consist of products, services, or functionality.

[2] Command, Control, Communications, Information Surveillance, and Reconnaissance (C4ISR)

[3] Theatres of conflict are becoming increasingly characterised by greater dispersion and maneuverability of forces over wide geographic distances that may often extend deep into an adversary's territory.

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

for the anticipated military environment. This occurs in the following section.

2.1 Military communications attributes

Military communication systems are generally expected to operate in environments that may be intolerable for commercial system operation. Accordingly, attributes of the military communication system, or more importantly the *environment* it is expected to operate in, have been used to characterise the system, and also to often distinguish it from commercial systems. The sustained growth in commercial telecommunication sector, however, has seen a steady increase in the demand for more military oriented attributes to become associated with commercial services. Although these services still would not meet the attribute requirements in a stringent military environment, they still may satisfy requirements for the less stringent environments that are more likely to occur[4].

The military attributes to be considered for this paper include, but are not limited to, the following descriptions;

- Quality of Service (QoS)
- Mobility
- Survivability
- Security

Each of these attributes is detailed in the following sections.

2.1.1 QoS

Quality of service is defined for these systems by the following characteristics.

- Data transfer rate
- Bit error rate, voice quality
- End-end propagation delay
- Call setup time
- Dropped call rate
- Capacity
- Communications reliability

The combination of these commercial characteristics help to define how effective a given communication service will be for C2 applications.

Comparisons to extant military communications systems such as tactical satellite communications and High Frequency (HF) show the new systems offer significant improvement in most of these measures of QoS. This is primarily due to improved source and channel coding and in response to increased demands from the commercial marketplace.

2.1.2 Mobility

Mobility is affected by the size of the terminal, the ability to establish and maintain communications on the move and the need for external sources of power. The advent of small battery powered computer such as PDA (Personal Data Assistants) and the integration with mobile communications provides numerous opportunities for situational awareness and command support systems for highly mobile forces.

Terminals requiring directional antennas tend to provide less mobility for small platforms (including troops) due to the complexity of acquisition and tracking systems.

In modern networks mobility may also include the ability to roam between networks, ie. Networks have the ability to mutually authenticate users and allow access. In this case service mobility may provide diversity and improve survivability of the service.

2.1.3 Survivability

Service survivability is a primary concern for military operations. Failure or loss of availability of a communications system in conflict can have disastrous consequences and the ability of a system to survive a range of incidental and deliberate actions is an important consideration for military planners.

Survivability can be thought of as security of supply and will be affected by the robustness and resilience of a system to a range of attacks, including factors such as congestion. Survivability must include considerations for conventional and non-conventional attacks on the air interface and on the supporting infrastructure used by each system to terminate or deliver traffic.

2.1.4 Security

Security refers to the ability to protect traffic (messages, voice and data) and traffic flow information. Because of the inherent mobility offered by most of the systems considered in this paper protecting traffic flow information may become highly important (ie. who is calling who, for how long, how often and from where).

Security is also defined to include protective measures to prevent deception through masquerading (pretending to be another user) and spoofing (injection of false messages).

All of the systems described in this paper claim to be developing security devices that offer varying degrees of end to end traffic security. The Inmarsat voice services all claim to be capable of supporting STU-III encryption. The newer systems aiming for US government markets (Iridium, Globalstar and ICO) are all developing Type-1 security products, mainly based on the FNBDT specification. This development will probably also see Type-3 and Type-4 encryption devices produced for the commercial marketplace.

[4] Note that the importance of each attribute is expected to vary according to the type of military scenario being considered.

3. Comparison of military and commercial markets

The differences between commercial and military communication attributes have been identified in the previous section, however, there still remains the problem of achieving a solution that will satisfy requirements and be cost effective in the long term. The type of solution will depend on the military planner's ability to exploit features of the commercial market, and also their willingness to modify these features according to their own market driven requirements. Furthermore, in order for this solution to be successful an understanding of the differences that exist between commercial and military products and services markets are required.
Some of these differences are outlined as follows;

- Commercial markets tend to be characterised by a more diverse, and much larger, number of users, which results in more mature products[5] that offer a wide range of features[6].
- The ability of an individual user to discard or upgrade a product or service is easier than it is for the military, which requires a "fleet" approach to procurement and maintaining compatibility between various upgrades of equipment or services[7].
- Product or service standards tend to be only valuable if they are popular, otherwise a defacto standard[8] occurs. This result is most likely to be associated with a commercial market.
- Commercial business models tend to adapt more rapidly to changing technologies than do military business models, i.e. doctrines.
- The pace of the changing commercial marketplace is faster than the military market, as evident by the much shorter life cycles of commercial products and services in comparison to those of the military[9].

3.1 Growth in commercial communications market

Rapid consumer uptake of mobile telecommunications, Figure 2, has resulted in great interest from telecommunications service providers in most countries of the world.

New companies focussed on service provision have spun off from traditional telecommunications companies and have experienced rapid growth in market capitalisation[10]. This increasing capitalisation results from rapid growth in consumer uptake compared to traditional fixed telephony services. Indeed some market forecasters are now predicting cellular telephony penetration rates of 300 percent (3 mobile phones per person) compared to 50 percent penetration of fixed services in developed countries.

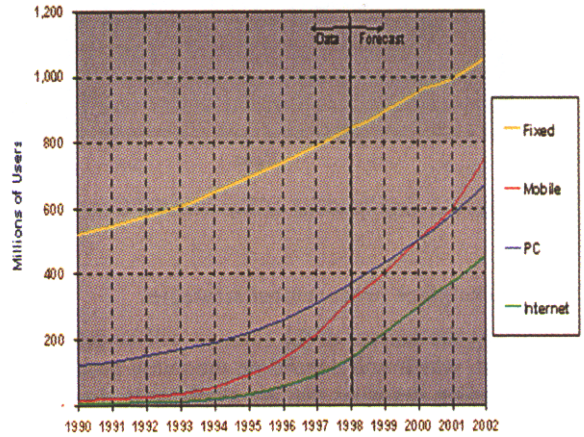


Figure 2: Growth in Commercial IT

To achieve these penetration rates implies that in the future the majority of cellular telephones will be used as embedded communications devices targeting machine to machine communications as well as for personal communicators. Furthermore, the rapid technological development and the enabling R&D investment can no longer be matched by military organisations and as a result Defence organisations risk being left behind unless it leverages the technological development in the commercial telecommunications sector.

Figure 3 shows one implication for military capability development if rapid commercial growth is not recognised as a factor. The axis marked Capability can represent any of the attributes listed in the previous section. This rapid advancement in the commercial sector may lead to an increasing technology deficit using a traditional military acquisition process. A better approach may be to adopt commercial technology in a way that adds military value yet maintain access to the commercial evolution path.

[5] A larger and more diverse number of users tends to accelerate the products "settling in" period.

[6] The large diversity in the types of users require a large range of features in order to meet the majority of requirements for these users.

[7] This would emphasise the importance of backward compatibility for military customers operating in a commercial market.

[8] Examples of defacto standards include, Windows operating system and software, and TCP/IP.

[9] This is largely due to the "fleet" procurement practice associated with the military acquisition. This acquisition approach, however is expected to change under Privately Funded Initiative (PFI) schemes, which may see a reduction in military product life cycles.

[10] For example, Japan's NTT DoCoMo is now capitalised at $527 billion dollars compared to $370 billion for its *parent* NTT. (Source: *Asia's mobile offspring dwarf their parents*, The Australian, 10/2/00, p 21)

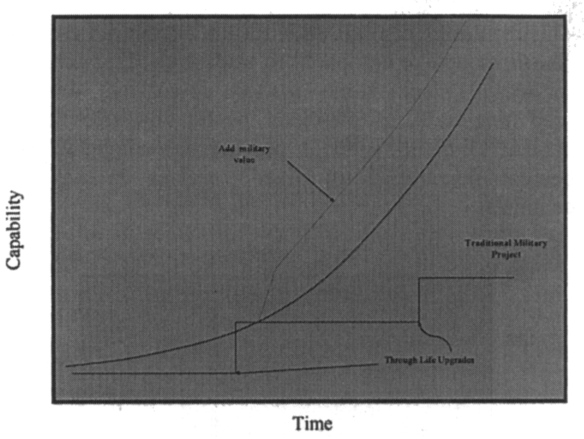


Figure 3: Implications of Exponential Growth

3.2 Marketplace procurement strategies

Ultimately, the success of the solution will be determined by its cost, which will depend on the initial procurement and upgrading strategies that are available in the military and commercial marketplaces.

A typical military investment strategy would be to provide significant up front capital investment in a product or service, as opposed to leasing commercial products or services at lower initial costs as shown in Figure 4.

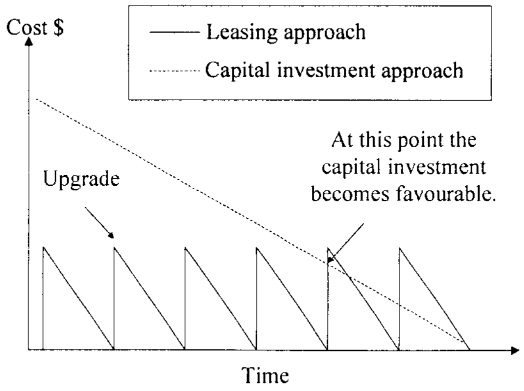


Figure 4 Leased versus capital investment strategies

Over time, the cost of leasing may eventually become higher than the dedicated military investment approach, however, access to the latest technologies and services, has been maintained over the entire period[11].

4. Application of COTS solutions for military communications

The challenge for military planners and capability development organisations is to identify those operational requirements that need specialised military communication services, and which can use military enhanced commercial services or unmodified services to meet these requirements.

They may also have to deal with the potential use by an adversary of the same advanced technology and devise methods of maintaining information superiority.

Some examples highlighting the application of COTS solutions for military communications are given in the following section.

4.1 Command and Control Warfare

One of the key challenges for future Command and Control Warfare (C2W), involving commercial communications systems, revolves around the ability to target particular users or groups of users. In many cases, both sides of a conflict will be using the same commercial communications networks. This will also be true of independent observers of conflict such as the UN, aid agencies and the media. The use of traditional means of C2W (degradation and denial) could well be counterproductive in complex conflicts

The means effective C2W strategies will allow a force to target individual subscribers or groups of subscribers. This is not a trivial task as all of these systems employ complex protocols that randomly allocate network resources to maximise capacity. Military forces need to be careful that C2W strategies do not force an adversary into using a C2 system that removes any advantage.

Area denial strategies based on dumb jammers may be effective, especially in combination with the use of smart antennas to enable friendly forces to overcome the effect of denial. Small, battery powered jamming terminals could be effective for this function but would have limited endurance for control of large areas. Airborne systems would be more effective and could support operations for longer periods of time from a reasonable stand-off distance using a directive antenna.

Other techniques for C2W could involve use of capabilities designed into the networks for fraud prevention and legal intercept requirements.

Diplomatic efforts to convince operators to enforce prioritisation may provide another simple and effective selection procedure. Most of the new commercial communications systems provide prioritisation and preemption capabilities although standard commercial practice is to assign almost all users to the same priority level. Operators would generally be hesitant to use prioritisation and preemption due to the perception that such use would adversely affect profits.

Similarly, the systems based on GSM maintain white, grey and black list of subscriber equipment. These lists are used by the commercial operators to manage fraud,

[11] Note that "break even" point will exist between the two approaches. Further note that if a customised solution were to be considered then the "serations" associated with the leasing strategy may begin to grow as the customised product begins to significantly deviate from the commercial product.

eg. the use of stolen phones or network access by a customer who does not pay the bill.

The white list contains details of subscribers who are entitled to have unrestricted access to the network. The grey list contains subscribers that may be of concern and use of the network by these subscribers generates an alarm in the network operations centre. The black list contains subscribers who are barred from using the network. These subscriber list can be readily changed by the operator and in a conflict could be used to warn of prevent network access by subscribers from a particular country or from a particular group. Operators may well seek compensation to allow such network management systems to be used as a means of C2W in a conflict.

4.2 COTS Communications in the ADF

While many nations are endeavouring to integrate COTS communications solutions into their military communication strategies, Australia has, through necessity , a considerable legacy in this area.

The Australian Defence Force currently employs a wide range of commercial communications technologies for the command and control of deployed forces. These services include leased capacity from civilian (Optus) and military (LEASAT) satellites to support broadband strategic, broadband tactical, and tactical mobile communications networks.

Australia also makes heavy use of Inmarsat and Intelsat services for off-shore deployments using commercial services with military encryption.

Further investigations into the potential application of commercial communication services are occurring through trials and evaluations in which the Defence Science and Technology Organisation (DSTO) participates in.

Current DSTO research programs are investigating the potential use of GSM, satellite PCS, broadcast technologies, ATM and IP communications products for tactical use.

Many of these systems were successfully deployed during the recent multilateral peace enforcement mission in East Timor.

5. Conclusion

New commercial satellite communications systems will be used for command and control in military conflicts of the future. The precedent was set in the Gulf war, when a large portion of military communications traffic was carried by commercial systems such as Inmarsat and Intelsat, and has followed in almost every major conflict since.

The advent of a number of systems offering new services and smaller, lower powered terminals may well create opportunities for well organised but less technically advanced adversaries to significantly improve their military capability.

The challenge is to ensure that these systems do not deliver an information advantage that could further complicate the range of military conflicts now experienced.

Similarly, as evidenced in Kosovo and East Timor, media and aid organisations using new communications technology have unprecedented mobility and this in turn could challenge military efforts in PSYCHOPS. Means of controlling benign parties such as media and aid agencies need to be devised. The danger is that some C2W techniques may inadvertently deny or degrade the communications of benign parties present in conflict scenarios and by doing so the results could well be counterproductive for the military effort.

The paper provides a brief summary of some issue related to military uptake of commercial communications systems. The focus has been on low data rate wireless technologies that might enable global command and control of national or multilateral forces. Some techniques for effective C2W against these systems based on inherent network capabilities and on external capabilities have been proposed.

Confidently Integrating COTS Software Under Worst Case Assumptions

Jeffrey Voas
Reliable Software Technologies
21351 Ridgetop Circle, Suite 400
Dulles, VA 20166, USA
jmvoas@rstcorp.com
Tel: 703.404.9293
Fax: 703.404.9295

Abstract

Most systems today are composed of hardware components, COTS software, and custom (bespoke) software. In terms of the software, the proportion of COTS software in a typical system is beginning to overtake the percentage of custom software. When a system fails, it may well be the COTS software that caused the system to fail given the well-publicized defect rates for acquired software. This paper describes a methodology for *predicting* the impact on system failure rates that a particular Commercial-Off-The-Shelf (COTS) software component might have before the component is embedded into the system.

1 Introduction

As software quality and information security becomes an increasingly well-publicized concern, the need for techniques that can accurately predict future failures and detect deficiencies grows. Voices from both industry and government are echoing this.

As an example, consider the comments of the US Department of Defense's CIO, Mr. Money, (June 17, 1999 issue of Federal Computer Week):

> *"The quality of software we are getting today is crap. Vendors are not building quality in. We are finding holes in it."*

Gary Beach, publisher of CIO Magazine, wrote on April 1, 1999:

> *"Are you tired of software vendors sending you service packs to fix bugs that should have been stamped out earlier? Off-the-shelf commercial software isn't good enough anymore. Service packs, indeed! Many CIOs I've talked to call them disservice packs."*

In my opinion, the underlying tension causing such comments to be made is directly related to the average defect density for all commercial software packages. According to Les Hatton,

> *"The industry standard for good commercial software is around 6 defects per KLOC in an overall range of around 6-30 defects per KLOC."*

Surprisingly, this rate has held fairly constant for the last two decades, regardless of the shift to object-oriented technology, automated debuggers, better test tools, stronger type safety in languages such as JAVA and ADA, etc.

If this range is correct, and given that COTS software is delivered in executable format (thus disallowing consumers to apply *white-box* techniques to assess for themselves the quality of the software) to the end users and system integrators, can the systems that rely on COTS software ever be trusted?

I will argue that the answer to this dilemma is "sometimes." And I will argue that even if the defect rate were higher than 30. While this is counterintuitive, the reason is that not all defects cause failure modes that are intolerable to the system.

The key then is to be able to predict, on a system-by-system basis, how well a system will be able to

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

tolerate COTS failures. This technique can also reveal what COTS failure modes the system will be able to tolerate.

To do so we employ a technique called Interface Propagation Analysis (IPA). IPA is a fault injection-based technique that simulates component and subsystem failures.

Our approach is simple. Start by simulating COTS component failures during system execution and observe how they affect the full system. If the effect is negligible, then it is fair to assume that if the component truly fails, the system will be able to tolerate real failures. If the impact is large, then the component needs additional scrutiny. The bottom line is that we do not care how poorly subsystems behave as long as their behaviors do not jeopardize the integrity of the full system.

As examples of the types of component failures that we might wish to simulate, consider events such as the COTS component hanging or failing to return a result to the system. Or it might be that the COTS component requires more memory than available and the component aborts.

IPA is normally applied once the software system is completed, thus it is a late life-cycle approach. However the analysis can still be applied before COTS components are integrated into the system provided that there exists a specification for what the component does such that we can generate failure modes from that specification. (Components that do not yet exist are termed "phantom components").

2 COTS and National Security

COTS systems cause great dependability fears. Probably nowhere is the concern greater than to information system security. The US Government considers the reliance of our military and national information infrastructure on public systems (such as the Internet and the telephone system) as severely compromising to national security. Currently, the US Government is spending billions of dollars in search of solutions to this vulnerability [1].

Biological systems use genetic diversity to enhance their survival. Each individual of a species is slightly different from another individual. The diseases that one individual is susceptible to may not damage another. This diversity increases the probability that a species will not be completely wiped out when epidemics occur. In information systems, however, we see the reverse trend occurring. We see less and less diversity being available, particularly in operating systems, due to the mainstream cry for standards and interoperability. In operating systems, we are converging towards two main platforms: UNIX and Windows. Operating systems are probably the most important of all COTS components today. Further, we are converging toward a handful of Web browsers, and this number, too, is likely to get smaller in the coming years. Because of this lack of diversity, we are all susceptible to the same types of attacks and vulnerabilities. And because our operating systems are off-the-shelf, we may also be

deficient in knowing everything going on in them and hence taking the appropriate action to protect ourselves.

The issue here is the *covert channel problem*. An executable component (other than the OS) may be making calls to the operating system that it is not supposed (and known) to. To determine whether this is happening requires a watchdog utility that has access to operating system level functionality. Tracking global environmental events requires the ability to keep track of the entire system. For example, it will probably be useful to monitor DeviceIoControl function calls. Not only will such calls need to be tracked, but isolating exactly who (or what component) is doing the calling is also required. This approach amounts to trying to wrap the operating system in order to see every request that enters or leaves the operating system. The downside to this approach is that it is both expensive to develop the utility, and expensive to execute it when the operating system is deployed. Also, this scheme would need to be implemented for each unique operating system.

3 Assessing COTS Software Failure Impacts

The first step in our approach is to determine how the system reacts to corrupted information being passed to it from COTS software functions. After all, if a COTS failure does not negatively impact the system, then concern over the dependability of the COTS component may be unwarranted.

As mentioned, the technique used here is *Interface Propagation Analysis* (IPA). The process of performing IPA is quite simple. The interfaces that are responsible for sending information out of a component to the system are first isolated. Random data generators are placed at those interfaces. As information exits a component, the generators grab the information and corrupt (modify) it. That modified information is then handed over to the system in place of the original information. This provides an analysis of how badly the system behaves when artificially corrupted information is injected into the state of the system.

One might wonder why we go through such an elaborate system to see how component failures affect the system. After all, why not just embed the components in and perform system-level testing? System-level testing will, in theory, determine this if component failures are frequent or the amount of system level testing is enormous. But if component failures are rare and the amount of system-level testing is limited, it is unlikely that system-level testing will provide any insight. So by forcing artificial component failures to occur, we can more quickly assess the tolerance of the system, even though we must always caveat our results with the realization that our injected failures were artificial.

IPA is composed of two software fault injection algorithms: "Propagation From" (PF) and "Propagation Across" (PA). PF corrupts the data exiting a real component (or phantom component) and observes what it does to the remainder of the system (i.e., what type of system failures ensue, if any). PF can also observe whether other subsystems fail and how. Thus, PF is an advanced testing technique that provides the raw information needed to measure the semantic interactions between components in order to measure their tolerance to one another.

PA corrupts the data entering a component. This process simulates the failure of system components that feed information into the component in order to see how it reacts. These simulated failures mimic human operator errors, failures from hardware devices, or failures from other software subsystems. After the component under analysis is forced to receive corrupt input, PA observes whether the component chokes on the bad data and fails. Note that PA is very similar to PF. The only difference is scale: PA is focused on standalone components and PF is focused on component/system interactions.

In summary, the main applications of IPA are: (1) making "buy" vs. "build" decisions, (2) recommending system redesigns when certain COTS failure modes have been demonstrated to be intolerable, and (3) providing intelligent heuristics for allocating testing resources. IPA provides information on how systems will tolerate the most detrimental failure modes of commercial software packages, hardware subsystems, and human operator errors. Here, for brevity, we have only focused on IPA simulating COTS component failures. (Further information on IPA can be found in [3,4,5].) By determining that even the worst failures from a COTS package are tolerable, the package can become a viable candidate for integration into the system even if it is relatively high in defects.

4 Summary

This paper has recommended methods for assessing whether a system can tolerate failures originating from COTS software subsystems. Because COTS software is often failure-prone, “defensive system designing" is prudent, and this paper has proposed one method that partially addresses this problem.

References

[1] General J. J. Sheehan. A commander-in-chief's view of rear-area, home-front vulnerabilities and support options. In *Proceedings of the Fifth InfoWarCon*, September 1996.

[2] J. Voas. Error Propagation Analysis for COTS Systems. *IEEE Computing and Control Engineering Journal*, 8(6):269-272, December 1997.

[3] J. Voas, F. Charron, G. McGraw, K. Miller & M. Friedman. *"Predicting How Badly 'Good' Software Can Behave"*. IEEE Software, 14 (4): 73-83, July, 1997

[4] J. Voas. *“Certifying Off-the-Shelf Software Components,"* IEEE Software, 31 (6): 53-59, June, 1998.

[5] A. Ghosh & J. Voas. "Innoculating Software for Survivability," Communications of the ACM, 42 (7): 38-44, July, 1999.

The Convergence of Military and Civil Approaches to Information Security?

(February 2000)

Robert Rowlingson
(Principal Scientist)

Defence and Evaluation Research Agency,
Woodward Building, DERA Malvern
St Andrews Road, Malvern
Worcs., WR14 3PS, UK

Introduction

The motivation for this paper is the about-turn that defence computing went through with open systems interconnection (OSI) and Ada. Defence specific products and bespoke development were discarded as the cost-benefits of mainstream COTS systems became far superior. This paper shows that a similar situation is developing in information security (infosec) and suggests that the defence approach to security may need to adapt if it is to benefit from the rapidly growing commercial market.

Civil Trends in Information Security

In recent years, the civil market for information security technologies has grown dramatically. The increasing requirement for information security arises from the need to mitigate the risks involved with:

- **E- commerce**: use of the internet both for business-to-business (B2B) and business-to-consumer (B2C) sales;
- **Mobility**: support for tele-working and mobile users;
- **Extranets**: the need to establish rich inter-connectivity with business partners (as well as customers and suppliers);
- **Knowledge management**: the increasing recognition of the importance and value of information in the 'knowledge economy'.

These drivers are forcing IT departments to develop and implement security policies that go beyond the boundary of the company. Internal systems can no longer be isolated from the outside world. Critical sources of company information such as the data warehouse must be made available to staff, partners, customers, potential customers and collaborators [1]. The pressure is on all companies to exploit the internet and to accept the risk that connecting to a public network inevitably carries. Lack of security and trust are the greatest inhibitors of the commercial use of the internet [2, 3]. Everyone accepts the need to practice 'safe hex'!

Anecdotal evidence of the relative importance of various infosec issues is indicated by the 'Top 5 Information Security Concerns for Corporations in 1999' [4].

1) Ability of current infrastructure to support e-commerce activities;
2) Implementing remote access without compromising the security of the corporate network;
3) 'Insider' attacks against corporate systems;
4) The extension of networks to support business partner connections;
5) Encryption and key management technology for customer facing systems.

Consequently companies are developing security policies; purchasing products such as firewalls and intrusion detection systems; procuring services such as penetration testing and security auditing; and training their staff in information security. In short, they are doing many of the things that defence has considered as essential for many years. This is driving massive growth in the information security market - in 2001 the internet security business is predicted to be worth $7bn with an annual growth rate in the whole information security industry of 65% [5].

Defence Trends and Information Security

The trend towards more joint and coalition operations means the ability to federate systems is essential. There is also a growing need to work with Non-Governmental Organisations as well as within the Government Secure Intranet (GSI). The Smart Procurement Initiative (SPI) has highlighted the need to use electronic commerce and for close partnerships with industry. The doctrine of 'information superiority' has brought the role of information in military operations to the fore. All four commercial drivers: e-commerce, mobility, extranets and knowledge management, are thus also applicable to the defence sphere. There is a *prima facie* case that the civil approach to security is becoming increasingly relevant to defence.

Comparing Defence and Civil Infosec

It would be comforting to believe that the massive civil investment in infosec and developments in areas like cryptography mean effective security will become easier. However some commentators believe that it is likely to get worse before it gets better. It is safe to assume that all systems can be successfully attacked in some way – probably in a way unexpected by the designers and with unexpected consequences. Defenders have to defend every vulnerability, whereas attackers have only to find one weak spot. Complex software will always have bugs

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

and features that pre-dispose it to security vulnerabilities. Security is dynamic and both risks and counter-measures are evolving rapidly. What does this mean for the convergence of military and civil approaches to infosec?

The military's ability to exploit the market will depend on whether the nature and level of protection it requires for information will be supportable using civil products and services. The requirements will depend on the perceived risks that security will be compromised, and the impact that a compromise has on the organisation's mission - in the case of defence to defend, in the case of industry to make profits. In both cases there is a trade-off between the protection of information and the sharing of information. Sharing is deemed to have a benefit; protection a cost. Thus at the heart of information security is a cost-benefit analysis which we can understand using four basic components of information security:

- Information is subject to various forms of potential **compromise**. A compromise is a breakdown in information security. It occurs whenever some property of information that needs to be preserved, is lost; examples include loss of confidentiality, integrity, availability, utility and authenticity;
- There is a **risk** of a given loss taking place. A risk is the chance that a potential compromise will actually occur;
- A loss will have some **impact** on the organisation. Impact is the cost to the organisation caused by a security compromise;
- An organisation can protect its information from compromise, by controlling the ways it shares information, with the support of carefully selected security controls known as **counter-measures**. A counter-measure is any action taken to reduce the risk, or potential impact of, an information security compromise.

It is the objective of information security to apply appropriate, cost-effective, counter-measures in order to reduce, the risk and impact of compromise, without undue effect on system usability. There is no universal agreement over the exact meanings of many terms in information security, however these components can support a discussion of how the defence requirements compare with the civil sector.

Compromise

The principal security concern of defence is to preserve the confidentiality of its information. It is now clear that these concerns are shared by industry which is giving a much higher priority to confidentiality than it has in the past. Companies in the US are estimated to be losing $250Bn annually to information thieves. Over a 17 month period some 1100 documented incidents of intellectual property theft were identified worth an estimated $44Bn [5, 6]. Trusted insiders are widely acknowledged as the single greatest threat to corporate information.

Risk

In providing a secure system, it is imperative that security measures should be designed to counter the most likely and most damaging causes of compromise. These can be characterised by [7]:

- accidents caused by the legitimate users of the system;
- the actions of a traitor, i.e. a legitimate user betraying the trust placed in them;
- trojan horse software unwittingly invoked by a legitimate user, thereby allowing illegal users access to the legitimate business processes;
- someone exploiting an implementation flaw or weakness in the security system;
- legitimate users failing to follow security operating procedures.

Risk assessment is the starting point for all security decisions and goes hand-in-hand with assessment of the potential impact. However we cannot precisely quantify risk - it is a probability which depends on many unknowns and unknowables. Formal quantitative risk assessment is prone to errors.

Companies are good at assessing and taking business risks. They make money by accepting and managing risks better then their competitors and this includes infosec risks. They take a pragmatic approach to the cost-benefit analysis, preferring skilled judgement to risk minimisation or analytical risk assessment. They are unlikely to favour the rigorous, quantitative approach used in defence.

In business, IT is purchased first and foremost for a given business purpose; security is then retrofitted to the system. In contrast the defence approach to security is based on a prescribed set of engineering principles; an analysis of how close a given system conforms to those principles and an evaluated assurance level. These principles tacitly assume a 'clean sheet' that can be designed from the ground up with security in mind. 'Security by design' is acknowledged as the best way to achieve security but as an approach it is fundamentally different to the civil sector.

The defence approach also contains a 'standard' attitude to risk. This has the benefit of consistency and accountability. However, the use of a pre-defined set of rules, enforced at system installation, is a significant restraint on flexibility and responsiveness by system owners. In the worst case, users may feel that security is not their problem, as it has already been addressed by a separate group of staff responsible for accreditation. As there is no guarantee that security adopted at the beginning of a project is suitable at a later date, security must be considered throughout the lifetime of a system.

The causes of compromise described above are made up of threats and vulnerabilities.

Threat

A threat is an action or event that can cause an information compromise. There are five broad classes of threat agents – criminals, terrorists, employees, other outsiders (e.g. former employees) and competitors (the enemy). Threat agents are either internal or external to an organisation; structured (i.e. organised in some way) or unstructured. To pose a threat, someone must possess the Skills, Knowledge, Resources, Authority and Motives (SKRAM) necessary to cause a compromise [8].

An area where the threat (or SKRAM) is increasing rapidly is hacking. Hacking is seen as 'cool' and in some circumstances is profitable. It is also becoming more widespread, as hacking tools are published on the web and many people start to experiment with them. However most of these 'script kiddies' are easy to repulse by competent system managers. The tools they naively employ are also useful to system managers to test and protect their networks. For example, L0phtCrack[1] is a password guesser. One NT administrator found 85% of his office's passwords in 20 minutes, all but two in a day [9]. It can and should be used by defenders to check that users are choosing passwords which are difficult to guess. Another common hacking technique is known as packet sniffing - the providers of L0phtCrack have also published a packet sniffer detector (a stealth packet sniffer is also available!). Hackers may be threats but their techniques often provide potentially useful countermeasures.

Vulnerabilities

A vulnerability is an inherent weakness in a system that may allow a threat to cause an information compromise. The critical issue for defence is that if it uses COTS software, it exposes itself to vulnerabilities that are well-known to potential attackers. The alternative approach is to develop non-COTS software and rely on 'security by obscurity', in other words to assume that as mainstream users and hackers do not use the software they cannot find any vulnerabilities to exploit. Indeed, many vulnerabilities are found in COTS technologies because users find bugs and exploitable 'features', and many people (good and bad) look for such vulnerabilities.

COTS software is potentially more secure than 'obscured' software in three ways. Firstly, the vulnerabilities that are easiest to find can be found and fixed; secondly, there is a short window of opportunity for an attacker to target a given system because fixes are released quickly and widely notified; and lastly, the chance that a system has unknowingly been compromised is less, as someone else using the same system may discover it first). To make this approach work, system vulnerabilities need to be monitored and disseminated – the role of a CERT (Computer Emergency Response Team) and patches and work-arounds must be quickly applied - the role of a system manager. Thus the COTS approach applies throughout the lifecycle and is less 'all or nothing' than a system that has some dependency on security by obscurity.

If the security of a system depends on it's obscurity then if that obscurity is ever compromised, and we may not actually know if it is, all bets are off. Further problems arise as such systems are generally not designed for rapid patching and updates, as with COTS. In some cases if a vulnerability was ever found, it would be very difficult to correct in the field. Examples of the failure of 'security by obscurity' in the civil sector are commonplace:

- The U.S. digital cellular companies created their own proprietary cryptography; some algorithms were made public without their consent [10]; once public they were broken. Now the industry is considering public algorithms to replace them;
- The security of Digital Versatile Disks (DVD) relies on the confidentiality of the code that performs the decryption in a DVD player. However, unencrypted code was found and this enabled the system to be reverse-engineered and compromised[2]. The software to do this was then posted on the web [11].
- Microsoft introduced Point-to-Point Tunnelling protocol (PPTP) as its Virtual Private Network (VPN) technology (competing with the internet standard IPSEC that has gone through rigorous public review). Microsoft fielded PPTP in Windows NT and 95, and published their protocols. In 1998 various flaws were published and Microsoft quickly posted a series of fixes (which were evaluated and found improved, but still flawed) [10].

These events also suggest that if security vulnerabilities are found in the COTS it procures then defence has a good chance of having them corrected by suppliers. (This is completely different from the case of functionality, where defence has very little influence over suppliers). However it is necessary to make the vulnerability known to the supplier and this would clearly involve some risk. Indeed this could be compounded by the fact that many suppliers appear to respond much more quickly to flaws that are in the public domain rather than those that have been communicated privately. It can seriously tarnish their reputation if they are seen to respond slowly to a security problem in a security product.

[1] The word L0phtCrack incorporates several techniques for choosing difficult-to-guess passwords.

[2] The key size had been limited by US export law but a brute force attack was not used, nor was any encryption algorithm broken – failure was due to a reliance on the confidentiality of an algorithm.

Impact

Defence has a very effective and well-developed system for evaluating impact – protective-marking. The classification of a document is a direct indication of the impact its loss of confidentiality would have on national defence. However the civil sector does not see that a detailed and rigorous multi-level classification scheme can be cost-effective. There is rarely a clear business case to undertake and maintain an information classification regime and the necessary vetting of staff. Furthermore, classification is of little use in understanding the impact of a loss of integrity or availability [12]. This is despite the benefits that such an approach offers, such as a better understanding of where to invest security resources and a visible reminder to staff to take appropriate care. Defence cannot look to COTS for support for multi-level document labelling schemes although simple support for 'need-to-know' type labels such as commercial-in-confidence, may be of interest to the civil sector and thereby provide a market for an appropriate COTS product.

Countermeasures

Thanks to the game of 'catch-up' between infosec defence and attack, counter-measure technology is a fast-moving field. The size of the market, and more importantly, the size of the e-business market it enables, suggests that it will continue to evolve rapidly towards the corporate mainstream. Much of it is highly relevant to defence. This section illustrates the proliferation of COTS technologies [13, 14]:

Biometrics

Biometrics refers to the ability of an attribute like a fingerprint to uniquely identify an individual. Techniques such as voice identification; fingerprints; facial, retina and iris recognition; and hand geometry are being pursued. High demand and improving technology is causing a rapid drop in price for usable biometric technologies.

Encryption

Encryption technology is a rapidly developing area in the civil sector which, until the publication of the concept of public key cryptography in 1976 (Diffie-Hellman) was almost the sole preserve of the defence sector [15]. Nowadays, innovative products are emerging in applications such as secure email, e-commerce, internet banking, copyright protection in digital media, cellular telephony and the like. These include:

- Certigrams - 2-dimensional representations of encrypted information;
- Hushmail (a browser-based email system like Hotmail but where the email on the mail server and all interactions with it are encrypted);
- Elliptic curve cryptography - a form of public key cryptography with a smaller key size and faster implementations than other public key algorithms.

In the past, defence has focussed on providing strong encryption algorithms to prevent enemy decoding. The civil sector recognises that a secure system is only as secure as its weakest link. Most threats will not attempt to crack encryption using code-breaking – it is far easier to steal keys or bribe people. The critical civil sector requirement is therefore for secure cryptosystems. The civil sector is also developing strong algorithms such as the Advanced Encryption Standard (AES). This is an open competition and civil cryptographers are attempting to find flaws in the competitors.

Encryption is becoming something of a 'silver bullet' in the civil sector. Despite its obvious potential and diverse applicability there remain many drawbacks:

- A perfect cryptosystem is no more achievable than perfect security;
- Encrypting everything everywhere is complex and costly;
- There are a variety of disparate algorithms, approaches and products;
- Encryption does little to counter trojan horse attacks;
- An attacker can use encryption to hide stolen information, malicious code, etc.;
- Encryption still requires identification for secure transactions;
- Encryption is a complex application - users can easily make serious mistakes.

The most worrying aspect of the widespread use of cryptography is that it may actually give users, and system owners, a false sense of security.

Snoopware

Snoopware is a colloquialism for software which monitors user behaviour and communciations[3] such as email; keystrokes; time and date of activities; name of program being used etc.. Snoopware is the internal analogue of intrusion detection – it has the potential to detect traitors and spot how accidental compromises occur.

Content Checking

Content checkers analyse information or 'content' passing in or out of a system. For example, outbound checks for words in email which suggest information is classified; inbound checks to block executables. Companies are using this software extensively to protect against their liability for the actions of their employees, who might send libellous or discriminatory emails. The difficulty is that little information is sent in plain ASCII text and checkers need to decompose attachments in compressed, encrypted or obscure formats, to apply a content policy to as much data as possible.

[3] Clearly there are many privacy issues concerning the use of such software.

Firewalls

Firewalls are the mainstay of network security. They form the first line of defence against network based attack. The main purpose of a firewall is to police a network access policy by examining and evaluating network traffic as it passes between networks. This strategic position means that as well as keeping the bad guys out they also enable the right connections to be made, for example to support the secure mediation of e-commerce. Consequently they are designed to let information through as much as to keep it out, acting like traffic lights to the various network protocols.

Vulnerability Scanners

Vulnerability scanners are tools that test a system against a database of known vulnerabilities. They have emerged as a key hacker tool but are also important for a sound defence. Several tools have initially been published on the web and then commercialised. Recently they have been used to probe internet systems, such as web and mail servers, to find whether they were running software with known security vulnerabilities. The Internet Auditing Project scanned almost every internet server and found several hundred thousand vulnerabilities. [16]

Intrusion Detection

Intrusion detection tools monitor access, attempted access and other interactions within and between networks. Basically they attempt to identify possible malicious behaviour, for example: repeated password guessing or non-standard attempts to create new users. They may monitor network traffic in real-time, or analyse audit logs off-line. Most products on the market look for specific patterns in user activity and tend to be inflexible. Some systems are now attempting to use heuristics and artificial intelligence techniques to improve detection rates and reduce false alarms.

Malware Protection

Malware is the generic name for harmful software such as viruses and trojan horses. Most anti-virus tools work by recognising 'signatures' of known viruses and require regular updates of new virus signatures. This means they may overlook new viruses, as the 'Melissa' outbreak demonstrated. Other malware protection software includes integrity checkers, which check that the system configuration has not been altered, and release sanctions, which ensure information is only released with user approval.

Information Security Management

Technology *per se* is of little use in information security if not backed up by policies and well-managed processes. Several civil standards are relevant:

- A Code of Practice for Information Security Management (BS7799-1:1999) [17] - aims to provide common, best practice guidance to enable an organisation to implement appropriate information security, and to facilitate inter-company trading by providing confidence in the security of shared information. It is an ISO 9001-like system in that it requires an organisation to 'say' what it does and 'show' that it does it, without specifying what the actual processes should be;
- System Security Engineering Capability Maturity Model (CMM) from SEI - a variant of the well-known software engineering CMM;
- Guidelines for the Management of IT security (GMITS ISO/IEC 13335) - provides a basis for an organisation to develop and enhance its security architecture and a means to establish commonality between organisations.

What appears to be missing is some way to allow one organisation to 'know' how secure another one actually is. This might be met by a composable security system description. In the world of inter-connected e-business this is a gaping hole.

Conclusions

In areas of IT where defence and civil sector requirements have a significant degree of overlap, defence has been persuaded that in order to keep pace with technology developments it must adopt the civil sector approach. This paper has highlighted the trends demonstrating that information security is heading in this direction. This implies that COTS technologies will become the default for many defence information security applications and that the defence and civil sector approaches to information security will converge. Note that COTS are not universal solutions. The highest classified information required in defence has no counterparts in the civil sector and there is unlikely to be any alternative to restricting such information to paper or isolated, physically-protected systems. However, there are a number of difficult issues which defence now faces:

- *How to manage risk more pragmatically*: Pragmatic risk management requires the application of judgement. This can be supported by a coherent traceable argument from high level policy, through individual project requirements and on to design, implementation and operation. Since perfect security is impossible there is always a degree of residual risk. The current approach does not manage it well because, unlike the civil sector, it does not accept it exists.

- *How to reduce its dependence on security by obscurity*: Defence systems are being developed that have a significant element of reliance on security by obscurity. However, if the obscurity is ever compromised (and it may not be clear if it has) an attacker may find the one vulnerability required. Attempting to fix a significant vulnerability in the field may be totally impractical.

- *How to gain confidence in the software it procures without a large formal assurance overhead*: The concept of assurance is not widely accepted in the civil sector[4]. Evaluations do not generally find common vulnerabilities, such as buffer overflows [18] and denial of service attacks [19]. Vendors are not prepared to jeopardise time-to-market for assurance, given their customers do not request it. There is a limited range of evaluated products. Frequently, only parts of the security functionality are assured. Until these drawbacks are overcome, assurance is unlikely to have a major impact in the market. Governments may need to re-define their approach, for example by recognising that mass use of software confers a certain degree of assurance and that open source software allows the requisite code inspection.

- *How to manage security on the timescales of the civil sector*: Information security is not static. If a system is 'secure' today it probably isn't tomorrow. It is dangerous to assume that a system, and its mode of operation, can be accredited at installation time as 'secure'. The civil view is that systems must always be considered insecure and that continual monitoring and rapid patching is essential.

- *How to address document marking*: The civil sector does not currently see a business case for the use of multi-level security and related document labelling. It is highly unlikely that COTS products will emerge to fulfil defence requirements. There are several alternatives: make labelling software available as a toolkit to promote its integration into defence systems; mandate a government labelling product; use encryption to manage security levels throughout its systems; or rely on informal separation and user conformance.

- *How to manage secure systems federation*: The internet demonstrates the effectiveness of a distributed, rather than a centralised, approach to systems federation. Currently this works for a simple trust model, namely trust nothing that you do not control. There is likely to be a requirement for techniques to manage more complex systems federation and secure service mediation for e-business.

In some of these aspects, such as the protection of information at the highest levels and the use of protectively-marked documents, it is unlikely that defence will find commercial solutions. In other areas, such as assurance, careful consideration is required to manage the mismatch between the defence and civil approaches. Finally, in its approach to issues such as risk assessment, static security, and security by obscurity, there are no technical reasons why the civil approach could not be used.

[4] Case law could overturn this view by ruling that 'due care' requires the use of evaluated products. However this is not likely in the short term.

References

[1] Turning Security On Its Head, The Forrester Report V13 (2), January 1999

[2] Nov 12 1999: A joint survey from @dtech and Talk City "The World Wide Internet Opinion Survey", found 83 percent of respondents had made an impulse buy online... and 45 percent cited security as the main deterrent.

[3] Security Portal Nov 15th, http://www.securityportal.com

[4] Bellcore/GlobalIntegrity's SecureComm 98 conference

[5] Computer Security Issues and Trends, 1999 CSI/FBI Computer Crime and Security Survey.

[6] Who's stealing your information? Dorothy E. Denning, http://www.infosecuritymag.com/apr99/cover.htm

[7] Interim Security Domain Modelling Guidance, Version 2.1, October 1999, DERA/CIS/CIS3/CR990148, K J Hughes

[8] Fighting Computer Crime, A New Framework for Protecting Information. Donn B.Parker, John Wiley and sons, 1998 http://www.washingtonpost.com/wp-dyn/business/A18205-1999Nov3.html

[9] http://www.lopht.com

[10] Cryptogram, September 1999, http://www.counterpane.com

[11] Hackers Unlock DVD Code, November 4th, Washington Post,

[12] CSI Roundtable: experts discuss the role of data classification now and in the future. Richard Power, Computer Security Institute Quarterly, V14 (2)

[13] A Guide to Security Technologies – A Primer for IT Professionals, RSA security

[14] Information Protection Fundamentals, Thomas R. Peltier, http://www.gocsi.com/ip.htm

[15] Applied Cryptography, Bruce Schneier. Wiley 1998

[16] The Internet Auditing Project http://www.securityfocus.com/templates/forum_message.html?forum=2&head=32&id=32

[17] The Revised Version of BS7799 – So What's New? Chris Pounder, Computers and Security V18 (1999) pp307-311

[18] Study says "buffer overflow" is most common security bug, www.cnet.com

[19] Computer Security – What Should You Spend Your Money On? Keith Buzzard, Computers and Security V18 (1999) pp322-334

Dynamic Detection of Malicious Code in COTS Software

Martin Salois and Robert Charpentier
{Martin.Salois@drev.dnd.ca, Robert.Charpentier@drev.dnd.ca}

Defence Research Establishment Valcartier
2459 Pie XI Blvd. North
Val Bélair, Québec, Canada
G3J 1X5

April 2000

Abstract

COTS components are very attractive because they can substantially reduce development time and cost, but they pose significant security risks (e.g. backdoors, Trojan horses, time bombs, etc.).

These types of attack are not detected by standard virus detection utilities, which are essentially the only commercially available tools that work directly on binaries. This paper presents a dynamic approach that intends to address this problem.

The complexity of a real time-bomb attack that disables a program after a fixed period of time is shown. Building on this example, a method that works at the binary level and that could be used to facilitate the study of other time bombs — and hopefully of all types of malicious actions — is presented. This is the first step toward a fully automated tool to detect malicious actions in all their forms.

The method, which monitors processor instructions directly, is currently intended specifically for Windows NT running on an Intel processor. It could easily be extended to other platforms. This paper also discusses the possibility of using dynamic analysis techniques to overcome the inadequacy of the static methods.

Finally, a brief survey is presented of commercial tools that attempt to address this issue, considering where these products are today and what is needed to obtain a credible sense of security, as opposed to the often false sense offered by some commercial tools.

1 Introduction

COTS software has become the de facto standard in most organisations today. From management's point of view, it is often much more advantageous to buy certain products off-the-shelf than to develop them in-house. The final product is often cheaper both in time and in money. It is more robust and offers more features than what can be expected from in-house development, and it usually enjoys much better long-term support.

Unfortunately, an application that is developed by some other company — possibly in another country — can pose a serious security risk. Although the threat from viruses has been known for years and many potent commercial protection tools are available, other threats such as Trojan horses, time bombs, logic bombs, covert channels and so on are not as easily dealt with. Once they become known, virus detection tools can cover some of them, but the key is "become known": most detectors work only with known and already analysed threats. As will be shown, there are virtually no commercial tools offering a reasonable level of protection against unfamiliar attacks.

DREV initiated the MaliCOTS project in 1997 to address this situation. This paper first compares dynamic and static analysis techniques. Then some preliminary work on dynamic analysis is presented, focusing on time bombs. Last, a quick overview is given of some of the commercial tools currently available. A broader view of the project is presented in [1].

2 Static vs. Dynamic Analysis

Program analysis can be static or dynamic, or can use some other kind of technique that cannot clearly be classified as one or the other. This section takes a closer look at the advantages and disadvantages of static over dynamic analysis.

First, using static analysis allows malicious code to be detected without actually running the program; thus ensuring that any malicious actions discovered will never be executed. Also, static analysis can give a good idea of the program's behaviour, for all possible execution conditions. And there is no performance overhead associated with static analysis: after a single successful analysis, the program can run freely. But despite these beneficial properties, there are some inconveniences. The main drawback

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

to using static analysis is the undecidability of many interesting properties: they cannot be determined for all cases. Also, the analysed code needs not be the one that is actually run: changes can creep in between analysis and execution. The static analysis of source code is particularly vulnerable to this last difficulty, because the code must be compiled. Not only is there a possibility that a malevolent entity will modify the source code directly, but the language libraries used might be modified so that changes are not apparent.

Basically, dynamic analysis has the opposite pros and cons. One cannot detect malicious code dynamically before it is executed, give or take a few commands. For example, imagine a five-instruction sequence that, taken together, forms malicious code. An analysis tool might keep track of the last few instructions or use a list of suspicious instructions and be able to block the execution of the fifth command. However, this method could be rather limited on its own, because of the lack of a more global view. But dynamic analysis does not suffer from the undecidability characteristic of static analysis, because all run-time values are available or can be made available at any point in the program. Although dynamic analysis can have significant overhead in run-time performance, as compared to static analysis, in the end it has one major advantage: the analysed code is the code that actually runs, without any possibility of alteration.

Although some detection techniques cannot be clearly defined as static or dynamic, most are one or the other. Some innovative techniques, however, clearly use hybrid analysis. For example, Colby [3] proposes a way to define guards statically for loop expressions and to determine if they can be proven to be effective; if not, dynamic guards are inserted to be checked at run-time, when the boolean value of the expression can be computed.

It seems clear that static and dynamic techniques could very well be combined to ensure better success in the discovery of malicious code. A tool could do all that is possible with static analysis to identify vulnerable areas precisely and then use dynamic analysis to try to eliminate them. For example, the tool could pinpoint areas of code where it knows or can determine that static analysis will fail, and then concentrate on these segments using a dynamic method. Thus the overhead of a dynamic process running on top of a program could be greatly reduced, allowing better surveillance of an untrusted program without exceeding a tolerable level of intrusion.

3 Time Bomb Detection by Monitoring

Preliminary studies suggested the need to focus on a small subset of malicious code to begin with. Since a guinea pig was at hand — a time bomb in a program library that was being tested — it was decided to study time bombs more closely.

This section starts by giving a definition of a time bomb. Then the details of a time bomb case study are presented. Finally, all possible ways of getting the time in the Win32 subsystem are examined to outline a possible way to detect time bombs.

3.1 Definition

A *time bomb* is malicious code that is triggered in a program when a specific logical condition relating to time is met. "Time" here refers to the actual system time and date or a countdown in seconds, hours, days, or even months or years. Although it could be argued that limiting the number of executions (before declaring the expiration of evaluation software, for example) could be called a time bomb, in this analysis it is considered a logic bomb.

For the purposes of security and detection, it does not matter whether or not the time bomb was inserted intentionally. An unintentional time bomb can still compromise the system.

Typical examples of time bombs are time computations that prevent a program from working after x hours, minutes, days, etc. If this type of time bomb is used appropriately, perhaps to protect proprietary software, it is not really "malicious." Even so, the process is rarely done in a correct and standard way, as will be seen in the next section, so it is still considered an unacceptable risk.

Other time bombs include viruses that are launched at specific dates: one that wishes "Merry Christmas!" or that commemorates a special day.

Of course, many programs legitimately need to use time triggers. For example:

- Virus scanners that you can schedule to work every day at a specific time,
- Meeting schedulers that notify you of appointments,
- Automatic backup programs,
- Games that limit the time to finish a puzzle.

Deciding if a "time bomb" is malicious or not has been left out of present concerns, although possible ways of deciding automatically will be discussed later on. For the moment, our only interest is in locating them; automatic classification mechanisms are useless without tools for detection.

3.2 A Case Study

To get a feel of what a time bomb may look like in assembly code, along with its possibly great complexity, a real-world example — from a source that shall remain nameless — will be examined. It will simply be called *SoftBomb*.

SoftBomb is distributed as a DLL. A demo version is available that will only work for one month. Thus, in effect, there is a time bomb that detects that the same day in the next month has passed and triggers the stopping of the

Code Excerpt 1: A legitimate time bomb

```
//...
SYSTEMTIME systemTime;
GetSystemTime(&systemTime);
if( systemTime.wMonth > previousSystemTime.wMonth ){
    // Then do something
//...
```

executable. Since *SoftBomb* is a DLL, it cannot actually stop the execution; it sends an error message saying that the evaluation time has expired when you try to initialise it. For the sake of simplicity, we will continue to say that it "stops" execution.

A time bomb needs to get the system time from somewhere. As will be seen in the next subsection, there are many ways of doing this, even restricting our studies to legal Win32 methods. After obtaining the date/time, the time bomb will check it against an installation date that it stored somewhere safe — preferably a place unknown to the user so he cannot simply delete it. This last requirement is not actually part of the time bomb itself, so this paper will not explore the assembly details of how *SoftBomb* stores the installation date, only the general scheme and how it can be bad for user systems.

The idea behind a successful protection scheme is to make it as obscure and as irrational as possible. If it is done in the simplest and most sensible way, crackers will have an easy time breaking it. For example, say you want to know if the month has changed in a legitimate time bomb. You would simply proceed as is shown in Code Excerpt 1. However, this way of doing things would be too simple a protection scheme. As will be seen, *SoftBomb* is much more "clever."

Note that in Win32 systems, the time and the date travel around in the same structures most of the time. For instance, `GetSystemTime` gives both time and date. There is no function called `GetSystemDate`. Unless noted otherwise, the term time will be used to mean both time and date.

This subsection takes a look at how *SoftBomb* gets the system date, how it does multiple checks on it, and where the installation date is stored. As a bonus, for completeness and possible future use, a few pointers are given on how one might crack *SoftBomb*.

First it must be mentioned that the time bomb in *SoftBomb* is located in a particular function that the user must call to initialise the library before use. Since *SoftBomb* is a DLL, this simplifies the analysis a little because DLLs are meant to be used by other programs that are not supposed to know their inner workings. This means that they have clear-text names and clear-cut boundaries for functions. This fact allowed us to narrow the search to a small fraction of the whole DLL. This will not always be the case: a time bomb could be scattered over a much larger fragment of code.

3.2.1 Getting the Date

Since simply getting the date with the `GetSystemTime` function would be too obvious, *SoftBomb* uses another approach: it opens a file that it is certain to find in the main Windows NT directory, and gets the time of the last access to this file.

In this case, the file is `win.ini`. Since it is an essential configuration file for Windows NT, it is always present and, by opening it, *SoftBomb* updates its access time. Hence, it gets the system date after transforming it from a file-time structure to a system-time structure.

The actual code is shown in Code Excerpt 2 (note that all the assembly code in this section was obtained by using Sang Cho's powerful Windows Disassembler [2]). The comments after semicolons are inserted automatically by the disassembler, which identifies common Win32 API, even with some form of static def/use analysis as, for example, at line 29 where it knows that `ebp` contains the address of the function `CreateFile`, inserted at line 23. The comments in italics after two slashes were inserted manually after analysis, to ease comprehension and to explain what is going on in the lines that were skipped to save space. Reserved keywords for instructions are highlighted in bold. Line numbers are used because they are more convenient than memory addresses.

Looking at the code more closely, it can be seen on the first line that *SoftBomb* gets the system directory, which is `c:\winnt\system32`, and stores the address of this string in `ecx` (line 2), then in `eax` in the function called at line 5.

After that, it surreptitiously changes the string, character by character, until it becomes `c:\winnt\win.ini` (lines 9, 11, 16, 17, 18, 23, 25, 26), interlacing this change with the normal operations necessary for the next function call to the API `CreateFileA` function (line 28), which requires 7 parameters (the 7 preceding pushes). This function is used to open the existing file `win.ini`.

Then *SoftBomb* uses the functions `GetFileTime` and `FileTimeToSystemTime` to get the file's last access time and convert it into the desired system time format. *SoftBomb* now has the current system time and date to do with as it pleases.

3.2.2 Checking the Date

To be certain that a cracker could not simply change one jump instruction to crack it, *SoftBomb* checks the date two different ways. And then to be really sure, it checks again.

Code Excerpt 2: Getting the current date

```
:1F0017AD call ebp ;;jmp KERNEL32.GetSystemDirectoryA
:1F0017AF lea ecx, dword[esp+00000288] // "c:"winnt"system32" at address ecx
:1F0017B6 push 0000005C
:1F0017B8 push ecx
:1F0017B9 call 1F0056F0 // among other things, copies ecx in eax
:1F0017BE add esp, 00000008
:1F0017C1 test eax, eax
:1F0017C3 je 1F00195D
:1F0017C9 mov byte[eax+01], 57 // changes memory to "c:"winnt"Wystem32"
// ... The next 6 instructions are inc eax 6 times (so eax=eax+6).
:1F0017D3 mov byte[eax-04], 49 // changes memory to "c:"winnt"WIstem32"
:1F0017D7 inc eax
:1F0017D8 push 00000000 //hTemplateFile = NULL
:1F0017DA push 00000080 //dwFlagsAndAttributes = FILE'ATTRIBUTE'NORMAL
:1F0017DF push 00000003 //dwCreationDistribution = OPEN'EXISTING
:1F0017E1 mov byte[eax-04], 4E //changes memory to "c:"winnt"WINtem32"
:1F0017E5 mov byte[eax-03], 2E //changes memory to "c:"winnt"WIN.em32"
:1F0017E9 mov byte[eax-02], 49 //changes memory to "c:"winnt"WIN.Im32"
:1F0017ED lea ecx, dword[esp+00000294]
:1F0017F4 push 00000000 //lpSecurityAttributes = NULL
:1F0017F6 push 00000003 //dwShareMode
:1F0017F8 mov ebp, dword[1F013194]
:1F0017FE mov byte[eax-01], 4E // changes memory to "c:"winnt"WIN.IN32"
:1F001802 push 80000000 // dwDesiredAccess = GENERIC'READ
:1F001807 mov byte[eax], 49      // changes memory to "c:"winnt"WIN.INI2"
:1F00180A mov byte[eax+01], 00 // changes memory to "c:"winnt"WIN.INI"
:1F00180E push ecx        // lpFileName = "C:"winnt"WIN.INI"
:1F00180F call ebp ;;jmp KERNEL32.CreateFileA //(7 parameters=7 pushes)
//... Checks for errors. Loads the time in registers for the following pushes.
:1F001831 push eax // lpLastWriteTime
:1F001832 push ecx // lpLastAccessTime
:1F001833 push edx //lpCreationTime
:1F001834 push esi // hFile
:1F001835 call dword[1F013190] ;;jmp KERNEL32.GetFileTime// (4 params=4 pushes)
//... Checks for errors. Closes the file. Puts the file time in registers for following pushes.
:1F00185E push eax // lpSystemTime
:1F00185F push esi // *lpFileTime
:1F001860 call edi ;;jmp KERNEL32.FileTimeToSystemTime// (2 params=2 pushes)
// ... Checks for errors.
```

So, there are three different checkpoint that are performed one after the other, each using different logic to see if the expiration year, date, and day have been reached or to check if the date itself has been tampered with. Let us look at them more closely.

The first checkpoint is pretty simple. It is shown in Code Excerpt 3. In the first line the installation date is compared with the current date, as obtained in the previous subsection. If the installation year is higher than the current year, it assumes there is an error, reset its structures and checks again. If there is still an error, it goes to the second checkpoint. There is actually a bug in *SoftBomb* at line 3: if the year has changed, it stops initialising *SoftBomb*, giving an expiration message. Therefore, if you install *SoftBomb* on December 31st, 1998, it expires on January 1st, 1999! Otherwise, the check continues by verifying that the expiration month has not passed. If it has, it ends the execution.

The second and third checkpoints are somewhat less independent than the first. In fact, they could probably be considered as a single checkpoint since the second jumps into the third to complete some checks. However, for clarity, it is better to view them as two different phases.

In Code Excerpt 4, the second checkpoint first compares the install year with the current year (line 1). If they are not the same, it moves on to the third checkpoint (line 2). If they are equal it checks the install month against the current month (line 4) and, if they are not the same, moves on to verify that only one month has passed and that the same day in the next month has not yet passed (line 7 and lines 10-19). If it is still the same month, it checks to see if the date is correct — that the system has not gone back in time — (line 7), decides that *SoftBomb* has not expired, and finishes its initialisation (jump at line 8).

SoftBomb then enters its third and final checkpoint, shown in Code Excerpt 5. At this point, it knows that the

Code Excerpt 3: First checkpoint

```
1  :1F00186E cmp word[esp+16], ax //installation year, current year
2  :1F001873 ja 1F001897 //install year ¿ current year (error, double check)
3  :1F001875 jne 1F001A20 //stops if year has changed
4  :1F00187B mov eax, dword[esp+1A]
5  :1F00187F xor ecx, ecx
6  :1F001881 mov cx, word[esp+0000008A]
7  :1F001889 and eax, 0000FFFF
8  :1F00188E inc eax
9  :1F00188F cmp eax, ecx //install month+1 (expiration month), current month
10 :1F001891 jl 1F001A20 //stops if expiration month ¡ current month
```

Code Excerpt 4: Second checkpoint

```
1  :1F001983 cmp word[esp+16], ax //install year, current year
2  :1F001988 jne 1F0019C6 //if install != current, go to next checkpoint
3  :1F00198A mov ax, word[esp+26]
4  :1F00198F cmp word[esp+1A], ax //install month, current month
5  :1F001994 jne 1F0019A2 //if install != current, check day
6  :1F001996 mov ax, word[esp+2A]
7  :1F00199B cmp word[esp+18], ax //install day, current day
8  :1F0019A0 jbe 1F0019FC //same year & same month & install current, OK
9  --------- //Inserted by disassembler to indicate a block's ending/starting.
10 :1F0019A2 xor eax, eax
11 :1F0019A4 mov ecx, dword[esp+1A]
12 :1F0019A8 mov ax, word[esp+26]
13 :1F0019AD and ecx, 0000FFFF
14 :1F0019B3 sub eax, ecx //current month, install month
15 :1F0019B5 cmp eax, 00000001
16 :1F0019B8 jne 1F0019C6 //if difference != 1, then go to next checkpoint
17 :1F0019BA mov ax, word[esp+2A] //difference = 1, check if same day not reached
18 :1F0019BF cmp word[esp+18], ax //install, current
19 :1F0019C4 jae 1f0019fc //if same day next month not passed, ok
```

year has changed (actually, because of the previously mentioned bug, *SoftBomb* never gets here, but let us pretend it does). At lines 1, 2, and 3, it checks to see if the difference is only one year; if not, it stops. If the year difference is indeed only one, it moves on to check if the current month is January (lines 4 and 5) and if the current month is December (lines 6 and 7), the only possible situation for a one-month evaluation. If this is not the case, execution stops; if it is, one final verification is made to check that the expiration day has not passed. Finally, if all is clear, the program continues with its normal initialisation.

3.2.3 Storing the Date

In the previous subsection, the installation date was mentioned. But where does *SoftBomb* store the date on which it was installed? As already stated, this question is actually outside the scope of time-bomb detection. But the reader might be interested, so *SoftBomb*'s approach will be outlined here.

It was also previously mentioned that it can hardly be considered malicious for a company to try to protect its software but that the methods sometimes used can be quite malicious if they are not regular and standard. The following discussion supports this point.

Once again, when it comes to hiding information for a protection scheme, obscurity is the way to go. You want to hide the information as deeply as possible, in a place where the user will not look; or should he decide to look, where he will not find anything suspicious.

SoftBomb's protection scheme is cunning in this sense because it does nothing for the first few uses. It waits a random number of times before storing an installation date on the hard drive. And a careful or suspicious user monitoring the first few runs of *SoftBomb* to see if it is legitimate is unlikely to catch the suspicious write to the Registry — the Registry is where all of Windows NT's configurations are stored — because *SoftBomb* stores the installation date there using a key inconspicuously named `FontAttributes`. A key with this name would easily be overlooked, especially since it is placed in a region of the Registry where the configuration of the desktop is kept (registry path `HKEY_CURRENT_USER\Control Panel\desktop`). Among the legitimate keys stored at this place, there are `AutoEndTasks`, `Pattern`, `IconHorizontalSpacing`, `IconVerticalSpacing`, `TileWallPaper`, `Wallpaper`, and so on. It is easy to see why one called `FontAttributes` would not be looked at twice.

Code Excerpt 5: Third checkpoint

```
:1F0019D9  sub eax, ecx  // install year, current year
:1F0019DB  cmp eax, 00000001
:1F0019DE  jne 1F001A20  // if difference != 1 then stop
:1F0019E0  cmp word[esp+26], 0001  // current month, January (01)
:1F0019E6  jne 1F001A20  // if current month not January then stop
:1F0019E8  cmp word[esp+1a], 000C  // install month, December (c=12)
:1F0019EE  jne 1f001a20  // if install month not December then stop
:1F0019F0  mov ax, word[esp+2a]
:1F0019F5  cmp word[esp+18], ax  // install day, current day
:1F0019FA  jc 1F001A20  // (jc=jb) if install day passed then stop
// ... Continue with normal initialization.
```

Finally, another random number of executions after expiration, *SoftBomb* creates another Registry key in the same registry path, called `DragDelay`. The purpose of this key is not completely clear, but it seems to be a flag that indicates that *SoftBomb* has expired. Since it is not really part of the time bomb itself, it was not investigated further.

Now that it has been shown how a real-world software product hides the installation date, it is trivial to demonstrate how the activity could be bad for a system: if programs were to write to the Registry anywhere they please, without ever cleaning up behind them, a maintenance nightmare would result. Legitimate and correct programs have a difficult enough task cleaning up their own mess; we cannot have programs writing where they are not supposed to. Clearly, such behaviour is unacceptable.

3.2.4 Cracking It

Only one matter remains to conclude this case study: how could the time bomb in *SoftBomb* be circumvented? Although some might perceive such action as a bad thing — after all, cracking software products is probably illegal in most countries — this example is only an illustration. The results of the work could later be extended to protect a system against more serious threats. For instance, a virus could be stopped dead in its tracks simply by dynamically stopping the time bomb that triggers it.

So, how could a "cracker" crack *SoftBomb*? That is, how can one remove the protection? There are many possible solutions. The two most plausible ones are given here or, at least, the two more practical in our view:

- Systematically replace all instructions that jump to the end sequence with `noops` in order to avoid ever getting to the stopping code. This could be done statically with a hexadecimal editor, or dynamically, on the fly.
- Add a routine to *SoftBomb* that would execute at the beginning of the initialisation function. This "hook" would simply delete the two registry keys identified in the previous subsection and transfer control back to the normal flow of the function.

Either solution could be used by a dynamic protection tool to thwart the time bomb, but the first seems more direct and easier to implement. One must simply reverse the jumps at run-time, a simple enough task for anyone familiar with debuggers.

This concludes our case study. The following sections look at the various ways to get the system time and date.

3.3 How to Get the Time and Date

This section explores the many ways to get the system time and date in Windows NT via the Win32 subsystem. The authors do not pretend that the list is exhaustive: smart, malicious attackers will always come up with new approaches. Also, it would take many pages to illustrate all the possible ways that the research team was able to devise to get the system time. A simple list of the Win32 functions that can provide the time or date and of the functions that can modify or control the time or date in any way. There are many of them, with many parameters that control their behaviour, and many functions that perform essentially the same task have different implementations with different names: `CreateFile`, `CreateFileA`, and `CreateFileEx`, for example. Consider this a first step in the construction of a database of knowledge on malicious code, a subject we will return to.

First, let us consider functions that can give the time of day (or the date) directly or indirectly in combination. Code Excerpt 6 presents the signatures of these functions. The function names are in boldface to make it easier to spot them among the parameters; they are presented in alphabetical order.

For historical reasons, many formats for the date/time exist and are still available. For example, a date can be computed from a long integer containing the number of seconds since 1970, or it can be directly stored as `dd-mm-yyyy`, or `yyyy-mm-dd`, and so on. This explains in part the large number of functions that can give the time/date.

It has been seen that the combination of `CreateFile`, `GetFileTime`, and `FileTimeToSystemTime` can be used to find the date. Following the same pattern, one could, for instance, create a file in MS-DOS mode (which is provided for backward compatibility), do

Code Excerpt 6: Functions that can be used to get and compare time and/or date

```
1  LONG CompareFileTime(CONST FILETIME *lpFileTime1, CONST FILETIME *lpFileTime2);
2  HANDLE CreateFile(LPCTSTR lpFileName, DWORD dwDesiredAccess, DWORD dwShareMode,
3     LPSECURITY_ATTRIBUTES lpSecurityAttributes, DWORD dwCreationDistribution,
4     DWORD dwFlagsAndAttributes, HANDLE hTemplateFile);
5  BOOL DosDateTimeToFileTime(WORD wFatDate, WORD wFatTime, LPFILETIME lpFileTime);
6  BOOL FileTimeToDosDateTime(CONST FILETIME *lpFileTime, LPWORD lpFatDate, LPWORD lpFatTime);
7  BOOL FileTimeToLocalFileTime(CONST FILETIME *lpFileTime, LPFILETIME lpLocalFileTime);
8  BOOL FileTimeToSystemTime(CONST FILETIME *lpFileTime, LPSYSTEMTIME lpSystemTime);
9  HANDLE FindFirstFile(LPCTSTR lpFileName, LPWIN32_FIND_DATA lpFindFileData);
10 HANDLE FindFirstFileEx(LPCTSTR lpFileName, FINDEX_INFO_LEVELS fInfoLevelId,
11    LPVOID lpFindFileData, FINDEX_SEARCH_OPS fSearchOp, LPVOID lpSearchFilter,
12    DWORD dwAdditionalFlags);
13 BOOL FindNextFile(HANDLE hFindFile, LPWIN32_FIND_DATA lpFindFileData);
14 BOOL GetFileTime(HANDLE hFile, LPFILETIME lpCreationTime, LPFILETIME lpLastAccessTime,
15    LPFILETIME lpLastWriteTime);
16 VOID GetLocalTime(LPSYSTEMTIME lpSystemTime);
17 LONG GetMessageTime(VOID);
18 VOID GetSystemTime(LPSYSTEMTIME lpSystemTime);
19 VOID GetSystemTimeAsFileTime(LPFILETIME lpSystemTimeAsFileTime);
20 DWORD GetTickCount(VOID);
21 NET_API_STATUS NetRemoteTOD(LPTSTR UncServerName, LPBYTE *BufferPtr);
22 LONG RegEnumKeyEx(HKEY hKey, DWORD dwIndex, LPTSTR lpName, LPDWORD lpcbName,
23    LPDWORD lpReserved, LPTSTR lpClass, LPDWORD lpcbClass, PFILETIME lpftLastWriteTime);
24 LONG RegQueryInfoKey (HKEY hKey, LPTSTR lpClass, LPDWORD lpcbClass, LPDWORD lpReserved,
25    LPDWORD lpcSubKeys, LPDWORD lpcbMaxSubKeyLen, LPDWORD lpcbMaxClassLen,
26    LPDWORD lpcValues, LPDWORD lpcbMaxValueNameLen, LPDWORD lpcbMaxValueLen,
27    LPDWORD lpcbSecurityDescriptor, PFILETIME lpftLastWriteTime);
28 BOOL ReportEvent(HANDLE hEventLog, WORD wType, WORD wCategory, DWORD dwEventID,
29    PSID lpUserSid, WORD wNumStrings, DWORD dwDataSize, LPCTSTR *lpStrings,
30    LPVOID lpRawData);
31 BOOL SystemTimeToFileTime(CONST SYSTEMTIME *lpSystemTime, LPFILETIME lpFileTime);
32 BOOL SystemTimeToTzSpecificLocalTime(LPTIME_ZONE_INFORMATION lpTimeZoneInformation,
33    LPSYSTEMTIME lpUniversalTime, LPSYSTEMTIME lpLocalTime);
34 MMRESULT timeGetSystemTime(LPMMTIME pmmt, UINT cbmmt);
35 DWORD timeGetTime(VOID);
```

a GetFileTime, and then do a FileTimeToDosDateTime to get the date in a different format that could be converted to system time.

Remember that the idea for a malicious scheme is to confuse an eventual detection process. So instead of creating a file, one could write a null character to a known file or simply open it. We will not attempt to cover all such variations.

The functions in lines 9, 10, and 13 could be used to go through the system directory files to extract the most recent date. Since the files in this directory are accessed often, at least the date will almost certainly be correct, if not the time. Similarly, the registry functions (lines 22, 24) could be used to get the last access time of often-used registry keys.

Via the logging mechanisms, it is possible to know when Windows NT was started. In many installations, the machines are rebooted every day. If that is case, the date is available directly. If it is not the case, functions that give the elapsed time since the last reboot (lines 20, 34) could be used to calculate the current date and time. A program could send a message to itself and then get the date of the event, which is automatically recorded by the mechanism. Many Win32 executables handle incoming messages — mouse clicks and keyboard commands, for instance — this way, so an attacker could use the function GetMessageTime to get the elapsed time between the starting of Windows NT and the handling of the message.

If an attacker knows that his target system obtains time information from a network, he can use the "network remote time of day" function (line 21).

This concludes our survey of how to get the time and date. Now, let us look at two ways to set a time bomb that do not require the application itself to look at the time. The signatures for these functions are given in Code Excerpt 7.

If the target is on a network, one can simply ask the system to wake the executable code up at a given future time. Of course, one must assume that it will still be running then.

Similarly, in the next two functions if one knows that the system runs for extended periods of time, one can set up a timer that will "beep" at regular intervals: several days or even weeks.

Code Excerpt 7: Functions to set the system time, a file time or to set a timer

```
NET_API_STATUS NetScheduleJobAdd(LPWSTR Servername, LPBYTE Buffer, LPDWORD JobId);
UINT SetTimer(HWND hWnd, UINT nIDEvent, UINT uElapse, TIMERPROC lpTimerFunc);
BOOL SetWaitableTimer(HANDLE hTimer, LARGE_INTEGER *pDueTime, LONG lPeriod,
   PTIMERAPCROUTINE pfnCompletionRoutine, LPVOID lpArgToCompletionRoutine, BOOL fResume);
```

3.4 Monitoring for Time Bombs

Based on these examples, it is now possible to propose ways to detect a time bomb in an executable code. Let us look briefly at two possibilities:

- Hook — that is, "intercept and redirect" in Win32 terminology — all of the time-supplying functions that were enumerated in the previous section. Determine who calls them and watch the callers for anomalous behaviour.

- The detection tool itself can get the date and verify on the fly, at assembly instruction level, if any data that is equivalent to the date is used to determine the results of conditional jumps.

Once more, "time" here really means "time and date."

In the next two subsections, the pros and cons of these two semi-automatic approaches are explored, then a combination of the two is proposed for maximum benefit. The subsection concludes with possible ways to automate the process by the use of specifications.

3.4.1 Hooking the Time Functions

This approach requires a program to intercept all calls made to the functions enumerated in Subsection 3.3. Commercial and freeware programs that do this have been noted, so the task should not pose too great a technical difficulty.

This technique would be used in a certifying environment; i.e., a closed and clean environment in which to perform extensive tests on the target program. During these tests, if the executable calls a "time" function, a flag is raised to look more carefully at the program to see if its behaviour has changed from normal. If it has, the tool can pinpoint the region of code where the time was accessed from and, hopefully, indicate if there is indeed a time bomb at that point in the assembly code of the executable.

By itself, this method cannot actually stop a time bomb from being triggered; it can only indicate the possibility of triggering and narrow the region for a human search. However, the intrusion level is minimal and the method would not limit the number of tests that can be run.

Evidently, if an attacker can devise a way to access the system time and date that was not included, this method would be powerless to detect it.

3.4.2 Comparing with Current Time

In this method, a monitoring tool would be created that is similar to what Jeffery proposed in [5]. A full-blown virtual machine is not needed; only a way to control the execution of applications and the ability to examine (and possibly change) the target program's memory.

A specialised monitor is needed, one that gets the time and date for itself. Then it runs the target program, opcode by opcode, and checks to see if it uses data equivalent to the time or date to control the execution flow. If so, and if the tool is being used in a certifying environment, it raises a flag telling the test engineer where to check the code more carefully. If it is not being used in such an environment, all it could do is to stop the application at that point, warn the user, and wait for further instructions. Because assembler code could not be provided for the user to verify, the message would have to be much simpler.

This method is certainly more powerful than the preceding one because it includes it. Effectively, if the target program uses the time data after returning from one of the "time" functions, this method will catch it. This method is also much more intrusive than the other, and consequently would be much slower.

3.4.3 Combining the two

To thwart the second method, an ingenious attacker could simply add a fixed number to the day, month, and year. If he adds 10, for example, and the monitoring application knows that the date is "03-04-2000," it would not detect control-flow jumps that check against 13, 14, and 2010 respectively. It would think they are simply numbers that the target program uses for its normal procedures. This was illustrated in the first checkpoint of *SoftBomb* example (Code Excerpt 3), where *SoftBomb* adds one to the installation month to know the expiration month. It could as easily have subtracted one from the current month to achieve the same result.

In order to prevent this simple scheme from defeating the second technique, it should be combined with the "hook" technique. Statically, it can recognise a call to a precise API function. It would be a simple matter to stop the target program only on "time" functions, and start examining the application closely only from there. This would considerably reduce the level of intrusion. A simple form of dynamic def/use graph could also be implemented to keep track of the time data to determine if a control flow condition is using some modified form of it.

To sum things up, a good way to detect a time bomb dynamically would be:

1. Create a monitor that can:
   - control the execution of a target program,
   - break on any instruction, and
   - examine the content of its memory address space.
2. Determine statically where the "time" functions are called and insert breakpoints at these points.
3. Execute the target program step-by-step, keeping track of time data and checking to see if the flow of control is influenced by it. If so, raise a warning.

The first step poses only technical difficulties, depending on the machine, the operating system and its architecture. The second step is even simpler since a good disassembler, such as the one that was used in Subsection 3.2, will do most of the job for us.

The last step is not that complex either. It only requires a good def/use mechanism to keep track of variables. This is easily done for registers, but problems may arise when memory is used to store variables and data structures. A resourceful attacker could use quite complex data structures, including recursive ones, or could even encrypt the time data. Nonetheless, building a def/use graph dynamically is a lot easier than doing it statically. The only major problem that can be foreseen is the amount of memory required to keep a "virtual double" of all time-related variables.

So far only a semi-automatic tool has been discussed: the first logical step toward a fully automated tool. First, knowledge needs to be gathered and a great deal of experimentation on the subject is required to augment our experience before our team can even think of automating the process. Still, if an automated tool is ever to see the light of day, it is necessary to tell the tool what is and what is not expected from a program. The following subsection addresses this subject.

Of course, static analysis could be combined with a dynamic tool. In the MaliCOTS project, static analysis techniques to detect malicious code are under investigation. The current plan is to combine the power of the two types of analysis, since a preliminary study indicates that the shortcomings of one are the strengths of the other (Section 2).

3.4.4 Giving Specifications

Following the example of Ko's work in [6], specifications could be used to tell our detection tool what the normal behaviour of the target program is. There are three main ways to give a specification:

1. Specify exactly what the application does.
2. Specify what it can and cannot do in general.
3. Specify a suspected vulnerability.

The first choice is impractical for long programs because of the sheer length of the specification, since one must "reverse-specify" the application.

The third choice is much easier to use, but it lacks generality: too much detail about actual time bombs must be provided. Moreover, this approach is useless against new time bombs. This approach suffers from the shortcomings of virus detectors: it is effective only against known attacks.

The authors believe that the second choice is the way to go. In the particular case of time bombs, a specification might be extremely simple: should the application base any of its normal operations on the current time? Yes or no?

Of course, finer grain specifications are needed in the case where an application is required to use the time. The language should be able to specify that a program needs the time for one particular input only, and for no other. In a fully automated tool, the administrator should be able to tell the monitor that "If the user requests that particular action, then the application should be allowed to use the time. Otherwise, it should not." For example, in a virus detection tool, if the user requests a scan every day at 6 o'clock then the monitor should know that it is permissible for the application to check the time against 6 o'clock, and not raise a warning. In any other situation it should raise one.

Specifications could also be useful to organise our knowledge of malicious code. For instance, if a grammar to specify malicious code is defined, a tool could be devised that would not need to be recompiled simply to add new knowledge to it. It could have a separate database that would be checked dynamically.

The two levels of specification could (and probably should) be combined. For example, to simplify specification writing, there should be only one way of specifying "get the time." For example, let the GetSystemTime function be the one and only function to get the time in our user-level specification. Then the user could say something very simple like:

```
SYSTEMTIME systemTime;
IF( GetSystemTime(&systemTime) )
   THEN violation();
```

Internally, our monitoring application would look in its database where all the different possibilities of getting the time are specified, link them with the GetSystemTime specification, check for them, and raise a violation if any is used.

In the end, the user-level specification might be as simple as a checklist showing all the possible malicious actions our tool can detect. The user would need only to check the kind of malice he wants to be warned against.

3.5 Time Bomb Detection – Conclusion

In this chapter, the process of creating and using a time bomb was examined very closely via the example of the expiration scheme for *SoftBomb*. It has shown that, in assembly language, the process can be quite complex. The instructions required might be spread through a large part of the executable code.

Although in this particular case the limitations imposed on DLL coding forced all the malicious code to be in one function, we will not always be this lucky. In a normal application, the malicious code could be scattered around the entire executable file. For example, an intelligent programmer could do what *SoftBomb* does — change the string `system32` to the string `win.ini` — while remaining unnoticed, by altering one letter at a time in seven different functions. The activity would certainly be more difficult to spot. Only the attacker would know which functions to execute to get the wanted result. He could make the process even more complicated by specifying an order for the function calls. By adding simple checks, he could see to it that the malicious function would be executed only by a precise sequence of operations, in effect creating a trapdoor.

Many ways to get the time and date, or to set timers to execute a task at a particular time have been described. The list may not be exhaustive, but it constitutes a vital first step towards identifying all the possible ways of getting system time.

Several approaches were proposed for a tool to detect time bombs. Although not all have been tested experimentally and no fully working prototypes have been created, the authors feel that the ideas expressed in this chapter could be useful not only toward the detection of time bombs, but also toward the goal of detecting any other kind of malicious code. Of course, any such steps would require that the extensive analysis that was performed for time bombs be extended to other forms of malicious code. The authors think that such a tool could relatively easily be adapted to provide continuous protection, as opposed to being used only in a testing environment. Because many errors in computer systems are the result of user error, such a tool would certainly be valuable.

4 COTS against COTS

Three commercially available products that offer protection against malicious code were examined, concentrating on those that can work at the desktop level — since most COTS will be installed via a CD-ROM or an intranet — and on those that are specifically designed to block malicious code — thus excluding network intrusion detectors. Most of the products examined have sister versions that can work at the network level. Although the selection is by far not exhaustive, most of the other available products have the same basic functionalities. Plus, almost all of these tools work only on mobile code (Java, ActiveX, JavaScript...), with some offering very basic protection against COTS that does not come from the network (e.g. CD-ROM, diskettes). This is the case for two of the three presented.

Neeley [11] gives a more complete list of available products, along with a good overview of what is at stake when dealing with this sort of program. Missing from this list are newer products from companies such as Norton and McAfee. The list of potential products is growing very rapidly, most of them claiming that they are the "First Product to Offer Complete Protection for Web Users". It can be rather confusing to determine exactly what level of protection is provided by current products.

4.1 Classifying

Randall [12] roughly defines three approaches to security for personal PCs. Most products today combine them to offer a wide range of protection. The three are:

Personal Firewall (Blocking) A simple gatekeeper that allows the user to control what passes in and out of communication ports. This only blocks certain channels, without any form of content analysis, and is therefore highly efficient speed-wise. Most firewall vendors have a personal PC version available. eSafe Protect Desktop uses this technology to block communication ports.

Sandbox Popularised by Java, the Sandbox model encloses the application in a virtual environment in which it can cause no harm. eSafe Protect Desktop also uses this technology to prevent selected programs from accessing specifically enumerated resources. This approach appears promising, but "Because of the high potential for programming errors, 'the sandbox is almost a moot point. You can't count on the sandbox for security,' says Ted Julian, a senior analyst for Forrester Research International" [11].

Scanning Much like current virus scanners, the tool scans the mobile code before downloading and executing it to see if it contains potentially malicious actions. It if does, the code if prevented from reaching the system. This technique is quite hard on system performance. Finjan's SurfinShield and Trend Micro's PC-cillin 6 both use this technique.

Let us examine these products in a little more detail and then discuss their shortcomings.

4.2 The Tests

Three products were tested, to give an indication of what is available on the market. The test consisted of trying to run the following documented hostile applets or ActiveX controls:

	Hostile Applets	Tiny Killer App	Exploder	Runner	ActiveX Check	Spy
eSafe Protect Desktop	9/9 blocked	NB	B	NB	13/17 blocked	NB
Surfinshield Online	9/9 blocked	NB	B	B	13/17 blocked	NB
PC-cillin	9/9 blocked	NB	B	NB	13/17 blocked	NB

Table 1: Comparison of what the three products successfully blocked (B: Blocked, NB: Not Blocked)

LaDue's Collection of Increasingly Hostile Applets [7] 9 documented hostile applets.

Tiny Killer App(let) [9] A small applet that forces Netscape to cause an access violation, thereby killing the browser.

McLain's Exploder [10] Exploder is an ActiveX control that performs a clean shutdown of your computer.

McLain's Runner [10] Runner is an ActiveX control that demonstrates how to run an arbitrary program on the browser's machine.

Smith's ActiveX checks [13] Checks for vulnerabilities to 17 documented hostile ActiveX controls.

Tegosoft's Spy [4] An ActiveX control that demonstrates how it can intercept what the user types on his keyboard. When activated, it replaces every key one types in NotePad into the sequence of letters forming `www.tegosoft.com` — press any key, and w appears, press 16 random keys and the whole sequence appears, the next key begins a new line and it starts again.

The Java applets were tested on both Netscape and MS Internet Explorer, while the ActiveX controls work only in MS Internet Explorer.

The results of the tests are presented in Table 1. All the products perform quite well on known and documented mobile code attacks, but unfortunately it is easy to find an attack that defeats them, as indicated by the tiny killer applet that eludes all three products.

Another interesting detail is that Tegosoft's SmartLoader, the ActiveX control responsible for loading the Spy control, was blocked at first by SurfinShield. This is interesting because the control is signed and perfectly legitimate. This illustrates the fact that legitimate software can easily be considered illegitimate. The line is not clear between what is legitimate and what is not.

eSafe Protect Desktop 2.1 According to its advertising, Aladdin Knowledge System's product "is a cutting edge, personal Internet content security solution for individual PC users, at home or at work. eSafe Protect Desktop includes a patent-pending anti-vandal sandbox module, an advanced, ICSA-certified anti-virus scanner, a unique personal firewall module, and a comprehensive resource protection system." (`http://www.esafe.com/products22/products.html`).

It includes an interesting sandbox feature that can, for example, prevent all programs from modifying the desktop, or prevent a specific application from accessing certain directories. It works as a super Access Control Lists (ACL) in the sense that, in addition to normal ACLs functions, which restrict access based on users, it allows access to be restricted for individual programs. Although this feature was of great interest in theory, in reality it did not stop the installation of the annoying WinZip icon on the desktop (☺).

The interface is attractive, although rather complex, as is the case with most tools in this category. This is definitely not entry-level material and, contrary to the publicity, it is not usable by the average user. As is so often the case, the default options do not offer the best level of protection the program can provide, which can be misleading.

The product provides full antivirus protection and it also creates and manages file integrity checks. Overall, it is a good contender and it is worth following up future versions.

Surfinshield Online 4.7 Finjan Software's product "enables companies to conduct e-business safely by providing proactive, run-time monitoring of executables, Java and ActiveX on corporate PCs" (`http://www.finjan.com/products_home.cfm`).

It uses a central server holding security policies and central knowledge. When a desktop detects a security breach, it informs the server, which immediately informs all clients, providing immediate protection for the entire network as soon as a breach occurs. Only the client is provided in the online version, the one tested; the server resides at Finjan's. Although this configuration limits options, it was used to provide a fair comparison with the other products.

A disturbing event occurs during installation: the product says that it is going to "adjust" your browsers. It is easy to understand that such a tool needs to make some changes to a system to protect it effectively. But what exactly does it do? Is the change safe? Does one really want a COTS product to change local programs?

LaDue [8] virulently describes the weaknesses of this product. In summary, he says that SurfinShield is only good at providing protection against known attacks. Even then, it is not very good since the "knowledge" is based on a list of URLs. LaDue's article is a bit dated and probably too rash — the product has definitely improved since the time of the judgement. But his drastic comments are indicative of shortcomings of all products currently on the market. Many of the general inadequacies common to most of these security products are discussed in the next subsection.

The product does not have antivirus protection; a separate tool is needed.

An interesting feature — once again, at least on paper, — is the SafeZone, which monitors the execution of a binary program. It is launched automatically on programs that come from the net and it can be launched manually to monitor a specific program. It stops a program from reading or writing files, making network connections, writing to the registry, or starting other programs. This works fine except that, frankly, what useful programs can one run under such constraints? This example illustrates a key concept in security: usability versus security.

PC-cillin 6.07 Trend Micro advertises this product as "all the protection you need to face the new Internet frontier!" (`http://www.antivirus.com/pc-cillin/products.htm`).

It is a typical example of new, emerging products. It is primarily an antivirus program that doubles as a malicious mobile code detector. As users become more aware of the security problems inherent to Internet use, they realise they need some form of protection. Companies see this opportunity and jump on it by offering their own products.

PC-cillin looks like a pretty good antivirus product — no tests were made of that use — but it is certainly lacking as a personal protection tool from the hazards of the Internet. Its single primary interface scans only incoming mobile code, much the same way that an antivirus program does. There are no facilities to protect from malicious files from CDs or an intranet — unless, of course, they contain viruses.

4.3 Shortcomings of COTS Desktop Security Products

First, because they are based on a priori knowledge of malicious code, they are unable to deal with unknown attacks. This is clearly not an acceptable approach since attackers will always be a step ahead of security tools. Furthermore, because commercial products of this nature are often rushed to delivery, they are quite error-prone. The problem is similar to that of current antivirus utilities, but more serious. It would be a full-time job for many users just to keep up with the patches and, given that in most cases the list of attacks must be updated manually, it is easy to understand that this is not a promising long-term solution. Future tools need to be able to detect suspicious behaviour on their own. Some form of "intelligence" is needed.

Second, they are usually quite complicated to use. Even though the actual level of customisation is rather limited, an expert is required most of the time, just to keep the product running without overpowering the routine activities of the system's users. Future tools need a powerful specification language for expert users and a very simple interface for everyday users. Then security administrators can set very precise policies and average users can be successfully protected without being annoyed by repetitive and often unspecific alert messages.

Along the same train of thought, the more tools one has or needs, the more confusion will be brought to the average user. For example, antivirus protection is a must for an organisation, as is protection from malicious code and intrusion. A perfect security tool would incorporate protection against all of these aspects in one package, providing the user with a single, consistent interface for all aspect of security.

Finally, most of the COTS Internet security products do not even attempt to address the problem of security in COTS obtained in executable format (e.g. MS Office, Eudora, MapObjects, and so on), which probably still account for the vast majority of purchased COTS. A complete tool obviously needs to be able to address the problems of binary programs.

5 Conclusion

Dynamic detection of malicious code has been outlined in this paper. This is one of the best techniques to detect malicious activity since it acts at the lowest possible level: processor instructions. Thus the MaliCOTS team concentrates its research effort on collaborative techniques that include both static and dynamic tools. It is our hope that dynamic analysis can complement static analysis and overcome its shortcomings. This will ensure the rigorous and efficient integration of COTS packages even when the source code is not available.

One of our top priorities at this time is to formalise the expression of security policy using a good specification language to discriminate malicious activities from acceptable behaviours. This requires very fine granularity. Currently, various design possibilities for a common security specification language are being examined within our research effort and a technology watch monitors commercial solutions.

Our team welcomes international collaboration.

References

[1] R. Charpentier and M. Salois. MaliCOTS:Detecting Malicious Code in COTS Sofware. In *Commercial Off-The-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"*, Neuilly-sur-Seine Cedex, France, Apr. 2000. NATO, RTO.

[2] S. Cho. Win32 Disassembler. `http://www.geocities.com/SiliconValley/Foothills/4078/`, Oct. 1998.

[3] C. Colby. *Semantics-based Program Analysis via Symbolic Composition of Transfer Relations*. PhD thesis, Carnegie Mellon University, Aug. 1996.

[4] T. Inc. Samples. http://www.tego.com/WebFrameSets/OcxControlKit/Samples.htm, 2000.

[5] C. L. Jeffery. A Framework for Monitoring Program Execution. Technical Report 93-21, University of Arizona, July 1993. Department of Computer Science, http://ringer.cs.utsa.edu/research/alamo/.

[6] C. C. W. Ko. *Execution Monitoring of Security-Critical Programs in a Distributed System : A Specification Based Approach.* PhD thesis, University of California Davis, Aug. 1996. Graduate Division.

[7] M. D. LaDue. A Collection of Increasingly Hostile Applets. http://www.rstcorp.com/hostile-applets/index.html.

[8] M. D. LaDue. The Rube Goldberg Approach to Java Security. http://www.rstcorp.com/hostile-applets/rube.html, 1998.

[9] G. McGraw and E. Felten. Java Security Hotlist. http://www.rstcorp.com/javasecurity/hotlist.html.

[10] F. McLain. ActiveX or How to Put Nuclear Bombs in Web Pages. http://www.halcyon.com/mclain/ActiveX/, 1997.

[11] D. Neeley. How to Keep Out Bad Characters. Security Management Online, 1998. http://www.securitymanagement.com/library/000599.html.

[12] N. Randall. Personal Security Suites. http://www8.zdnet.com/pcmag/features/personal_security/_open.htm, 1997.

[13] R. M. Smith. ActiveX Security Check Page. http://www.tiac.net/users/smiths/acctroj/axcheck.htm, 1999.

The Ruthless Pursuit of the Truth about COTS

Dr. Norman F. Schneidewind
Naval Postgraduate School
2822 Racoon Trail
Pebble Beach
California, 93953, USA
Email: nschneid@nps.navy.mil

Abstract

We expose some of the truths about COTS, discounting some exaggerated claims about the applicability of COTS, particularly with regard to using COTS in safety critical systems. Although we agree that COTS has great potential for reduced development and maintenance time and cost, we feel that the advocates of COTS have not adequately addressed some critical issues concerning reliability, maintainability, availability, requirements risk analysis, and cost. Thus we illuminate these issues, suggesting solutions in cases where solutions are feasible and leaving some questions unanswered because it appears that the questions cannot be answered due to the inherent limitations of COTS. These limitations are present because there is inadequate visibility and documentation of COTS components.

Introduction

In this paper we analyze three important aspects of COTS software: 1) reliability, maintainability, and availability; 2) requirements risk assessment, using risk factors from the Space Shuttle and modifying them for more general use; and 3) cost framework. We are motivated to address these issues because we feel that the COTS community has not adequately addressed some very important questions concerning the applicability of COTS when used in a host system. We define a host system as follows: it contains both COTS and non-COTS software; the latter is specific to the operational mission of the organization; and the mission cannot be satisfied entirely by COTS components. Our concerns are reinforced by Kohl: "The most significant challenges of V&V of COTS products has to do with knowledge of the functionality, performance and quality of these products. Because these products tend to be developed for large, commercial markets as opposed to being developed to a specification for a single customer, they tend to provide a variety of useful and desirable features for the market that they are targeted for, at the expense of the specific system needs in which such products may be used. Further, quality and reliability are sometimes not considered critical when time-to-market is a driving requirement. Thus, it is sometimes the case that these COTS products contain features and functionality that may not be fully known, even to the vendor." [KOH99].

Many vendors produce products that are not domain specific (e.g., network server) or have limited functionality (e.g., mobile phone). In contrast, many customers of COTS develop systems that are domain specific (e.g., target tracking system) and have great variability in functionality (e.g., corporate information system). This discussion takes the viewpoint of how the customer can ensure the quality of COTS components. In addition to direct quality evaluation, we also consider requirements risk analysis in a later section, which indirectly affects quality. We must distinguish between using a non-mission critical application like a spreadsheet program to produce a budget and a mission critical application like military strategic and tactical operations. Whereas customers will tolerate an occasional bug in the former, zero tolerance is the rule in the latter. We emphasize the latter because this is the arena where there are major unresolved problems in the application of COTS. Furthermore, COTS components may be embedded in host systems. These components must be reliable, maintainable, and available, and must interoperate with the host system in order for the customer to benefit from the advertised advantages of lower development and maintenance costs. Interestingly, when the claims of COTS advantages are closely examined, one

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

finds that to a great extent these COTS components consist of hardware and office products, not mission critical software [CLE97].

Obviously, COTS components are different from host components with respect to one or more of the following attributes: source, development paradigm, safety, reliability, maintainability, availability, security, and other attributes. However, the important question is whether they should be treated differently when deciding to deploy them for operational use; we suggest the answer is *no*. We use *reliability* as an example to justify our answer. In order to demonstrate its reliability, a COTS component must pass the same reliability evaluations as the host components, otherwise the COTS components will be the weakest link in the chain of components and will be the determinant of software system reliability. The challenge is that there will be less information available for evaluating COTS components than for host components but this does not mean we should despair and do nothing. Actually, there is a lot we can do even in the absence of documentation on COTS components because the customer will have information about how COTS components are to be used in the host system. To illustrate our approach, we will consider the reliability, maintainability, and availability (RMA) of COTS components as used in host systems.

In addition, COTS suppliers should consider increasing visibility into their products to assist customers in determining the components' fitness for use in a particular application. We offer ideas about information that would be useful to customers and what vendors might do to provide it.

This paper is organized as follows: reliability, maintainability, availability, requirements risk analysis, improved visibility into COTS, cost as the universal COTS metric, and conclusions.

Reliability

There are some intriguing questions concerning how to evaluate the reliability of COTS components that we will attempt to answer [SCH991]. Among these are the following: How do we estimate the reliability of COTS when there is no data available from the vendor? How do we estimate the reliability of COTS when it is embedded in a host system? How do we revise our reliability estimates once COTS has been upgraded? A fundamental problem arises in assessing the reliability of a software component: a software component will exhibit different reliability performance in different applications and environments. A COTS component may have a favorable reliability rating when operated in isolation but a poor one when integrated in a host system. What is needed is the operational profile of COTS components as integrated into the host system in order to provide some clues as to how to test COTS components. We will assume the worst-case situation that documentation and source code are not available. Thus, inspection would not be feasible and we would have to rely exclusively on testing and reliability calculations derived from test data to assess reliability.

The operational profile identifies the criticality of components and their duration and frequency of use. Establishing the operational profile leads to a strategy of what to test, with what intensity, and for what duration. We must recognize that a COTS component must be tested with respect to *both* its operational profile and the operational profile of the host system of which it is a part. The COTS component would be treated like a black box for testing purposes similar to a host component being delivered by design to testing but without the documentation. Testing the COTS components according to these operational profiles will produce failure data that can be used for two purposes: 1) make an empirical reliability assessment of COTS components in the environment of the host system and 2) provide data for estimating the parameters of a reliability model for predicting future reliability [SCH97].

A comprehensive software reliability engineering process is described in [ANS93]. As pointed out by Voas, black box and operational testing alone may be inadequate [VOA98]. In addition, he advocates using fault injection to corrupt one component (e.g., COTS component) to see how well other components (e.g., the host system) can tolerate the failed component. While this approach can identify problems in the software, it cannot fix them without documentation. Thus there must be a contract with the vendor that allows the customer to report problems to the vendor for their resolution. Unfortunately, from the customer's standpoint, vendors are unlikely to agree to such an arrangement unless the customer has significant leverage such as the Federal Government. In the case where documentation is available, it would be subjected to a formal inspection of its understandability and usability. If the documentation satisfies these criteria, it would be used as an aid to inspecting any source code that might be available. Next we consider COTS maintainability issues.

Maintainability

In the case of maintainability, there are more intriguing issues. Suppose a problem occurs in a host system. Is the problem in COTS or in the host software? Suppose it is caused by an interaction of the two. The customer knows the problem has occurred, but does not know how to fix it if there is no documentation. The vendor, not being on site, does not know the problem has occurred. Even the vendor may not know how to fix the problem if the source of the problem is the host software or an interaction between it and COTS components. In addition, suppose the customer needs to upgrade the host software and this upgrade is incompatible with the COTS components. Or, conversely, the vendor upgrades COTS components and they are no longer compatible with the host software. Lastly, suppose there are no incompatibilities, but the customer may be forced to install the latest COTS components upgrade in order to continue to receive support from the vendor. None of these situations can be resolved without either the customer having documentation to aid in fixing the problem, or a contract with the vendor of the type mentioned above. As in the case of reliability, when neither of these remedies is available, problems can only be identified but they cannot be fixed. Thus the software cannot be maintained. An additional factor that impacts both reliability and maintainability is that the vendor is unlikely to continue to support the software if the customer modifies it. Thus the situation degenerates to one in which the customer is totally dependent on vendor support to achieve reliability and maintainability objectives. This may be satisfactory for office product applications but it is unsatisfactory for mission critical applications. Next we consider the COTS availability issues.

Availability

High availability is crucial to the success of a mission critical system. What will be system availability using COTS? To attempt to answer this question, it is useful to consider hardware as a frame of reference. The ultimate COTS is hardware; it has interchangeable and replacement components. Maintenance costs are kept low and availability is kept high by replacing failed components with identical components. Unlike hardware, availability cannot be kept high by "replacing" the software. A failed component cannot be replaced because the replacement component would have the same fault as the failed component. Fault tolerant software is a possibility but it has had limited success. We see that availability is a function of reliability and maintainability as related by the formula:

Availability = MTTF/(MTTF+MTTR) =

1/1+(MTTR/MTTF),

where MTTF is mean time to failure and MTTR is mean time to repair. MTTF is related to reliability and MTTR is related to maintainability. For high availability, we want to drive *time to failure* to infinity and *repair time* to zero. However, we have seen from the discussion of reliability and maintainability that achieving these objectives is problematic. Thus to achieve high availability, either the COTS software must be of high intrinsic reliability – probably a naive assumption – or there must be in place a strong vendor maintenance program (this assumption may be equally naive). Next we consider COTS visibility issues.

Improved Visibility into COTS

Major drawbacks of including COTS in a software system are the lack of visibility into how the COTS components were developed and an incomplete understanding of the components' behavioral properties [SCH991]. Without this information, it is difficult to assess COTS components to determine their fitness for a particular application. As suggested by McDermid in [TAL98], a partial solution might be for COTS vendors to identify a set of behavioral properties that should be satisfied by the software, and then certifying that those properties are satisfied. For instance, an operating system supplier might certify that a lower-priority task does not interrupt a higher priority task as long as the higher priority task holds the resources required to continue processing. COTS vendors might also include the specifications of those components as well as details of verification activities in which those specifications had been used to show that specific behavioral properties of the software were satisfied. For instance, an effort in progress at the Jet Propulsion Laboratory [JPL98] involves developing libraries of reusable specifications for spacecraft software components using the PVS specification language [SRI98]. The developers of the libraries work cooperatively with anticipated customers to develop the specifications and identify those properties that the components should satisfy. As they develop the libraries, the component developers use the PVS theorem proverb to show that the behavioral properties are satisfied by the specification. These proofs are intended to be distributed with the libraries. When customers modify the libraries, perhaps to customize them for a new mission, they will be able to use the accompanying proofs as a basis for showing that the modified specification exhibits the desired behavioral properties. Similarly, commercial vendors could work with existing and potential customers through user groups to discover those behavioral properties in which users are the most interested, and then work to certify that their components satisfy those properties. Next we present a methodology for analyzing requirements risk when COTS is embedded in a host system.

Requirements Risk Analysis

In this section we first describe the Shuttle risk management process. Then we consider how it could be modified to accommodate the use of COTS. In providing this analysis, it should not be inferred that we necessarily advocate the use of COTS on the Shuttle or on any other safety critical system. Whether COTS should be employed would depend upon many environmental and application factors. Rather, our goal is to investigate whether the Shuttle risk analysis process is adaptable to the use of COTS.

Shuttle Risk Management Process

One of the software development and maintenance problems of the NASA Space Shuttle Flight Software organization is to evaluate the risk of implementing requirements changes. These changes can affect the reliability, availability and maintainability of the software. To assess the risk of change, a number of risk factors are used. The risk factors were identified by agreement between NASA and the development contractor based on assumptions about the risk involved in making changes to the software. This formal process is called a risk assessment. No requirements change is approved by the change control board without an accompanying risk assessment. During risk assessment, the development contractor will attempt to answer such questions as: "Is this change highly complex relative to other software changes that have been made on the Shuttle?" If this were the case, a high-risk value would be assigned for the complexity criterion. To date this qualitative risk assessment has proven useful for identifying possible risky requirements changes or, conversely, providing assurance that there are no unacceptable risks in making a change.

The following are the definitions of the risk factors, where we have placed the factors into categories and have provided our interpretation of the question the factor is designed to answer. In addition, we added the risk factor *requirements specifications techniques* because we feel that this one could represent the highest reliability risk of all the factors if a technique leads to misunderstanding of the intent of the requirements. For each of the risk factors, we analyze its appropriateness for COTS. As you will see, this analysis not only determines the adaptability of the process to COTS, but also exposes some serious issues in the employment of COTS in *any* system. For example, the Shuttle risk process is all about assessing the risk of requirements changes. In COTS, we would not want to attempt changes because we don't have the necessary source code and other documentation. Furthermore, if we did make a change, it could invalidate our software license. This situation illuminates a serious deficiency in using COTS. Therefore, our only recourse, if feasible, is to change the host software to reflect the change. In other words, COTS has to be used "*as is*" in our system. Thus, in what follows, the risk factors are a function of the change in the host software and how the change relates to and can be integrated with COTS.

In order to modify the Shuttle risk process to make it applicable to the use of COTS, we must change the software change metric from lines of code to components. In addition, we must change our view of the software from a set of individual instructions to a set of interconnected components. Otherwise, it would make no sense to talk about number of lines of code to be changed in the host software when we only have visibility of COTS at the component level. We will also assume an object oriented development and maintenance paradigm.

Requirements Change Risk Factors

The following are the definitions of the Shuttle risk factors modified to accommodate the use of COTS, where, as mentioned previously, only *host software components* can be changed, but in making the changes, the relationship with COTS components must be considered. If the answer to a yes/no question is "yes", it means this is a high-risk change with respect to the given factor. If the answer to a question that requires an estimate is an anomalous value, it means this is a high-risk change with respect to the given factor. When a change to a component is mentioned below, it will be understood to be a change to host software.

Complexity Factors

o Qualitative assessment of complexity of change (e.g., very complex)

- Is this change highly complex relative to other software changes that have been made on the system? What are the interfaces between the host components and COTS components that are affected by the change? Is the change more complex for the host system than for the host software alone?

o Number of modifications or iterations on the proposed change

- How many times must the change be modified or presented to the Change Control Board (CCB) before it is approved?

Size Factors

o Number and types of components affected by the change

- How many components and types of components must be changed to implement the requirements change?

o Size of software components that are affected by the change

- How many component objects are affected by the change?

Criticality of Change Factors

o Whether the software change is on a nominal or off-nominal component path (i.e., exception condition)

- Will a change to an off-nominal component path affect the reliability of the software?

o Operational phases affected by the changed component path (e.g., ascent, orbit, and landing)

- Will a change to a critical phase of the mission (e.g., ascent and landing) affect the reliability of the software?

Locality of Change Factors

o The area of the affected change (i.e., critical area such as a component path for a mission abort sequence)

- Will the change affect objects of components that are critical to mission success?

o Recent changes to components in the area affected by the requirements change

- Will successive changes to the components in a given area lead to non-maintainable code?

o New or existing components that are affected

- Will a change to new components (i.e., a change on top of a change) lead to non-maintainable software?

o Number of system or hardware failures that would have to occur before the components that implement the requirement are executed

- Will the change be on a component path where only a small number of system or hardware failures would have to occur before the changed components are executed ?

Requirements Issues and Function Factors o Number and types of other requirements affected by the given requirement change (requirements issues)

- Are there other requirements that are going to be affected by this change? If so, these requirements will have to be resolved before implementing the given requirement.

o Possible conflicts among requirements changes (requirements issues)

- Will this change conflict with other requirements changes (e.g., lead to conflicting operational scenarios)

o Number of principal software functions and components affected by the change

- How many major software functions and components will have to be changed to make the given change?

Performance Factors

o Amount of memory required to implement the change

- Will the change use memory to the extent that other functions and components will not have sufficient memory to operate effectively?

o Effect on CPU performance

- Will the change use CPU cycles to the extent that other functions and components will not have sufficient CPU capacity to operate effectively?

Personnel Resources Factors

o Number of inspections of components and objects required to approve the change

- Will the number and duration of inspections be significant?

o Manpower required to implement the change

- Will the manpower required to implement the software change be significant?

o Manpower required to verify and validate the correctness of the change

- Will the manpower required to verify and validate the software change be significant?

Tools Factor

o Software tools creation or modification required to implement the change

- Will the implementation of the change require the development and testing of new tools – for example the development of component and object testing tools?

o Requirements specifications techniques (e.g., flow diagram, state chart, pseudo code, control diagram).

- Will the requirements specification method be difficult to understand and translate into components and objects?

As an example, Table 1 shows a partial list of the risk factors compiled for the for the Shuttle *Three Engine Out Auto Contingency* and *Single Global Positioning System* requirements changes.

Table 1

Change Request Number	SLOC Changed	Complexity Rating of Change	Criticality of Change	Number of Principal Functions Affected	Number of Modifications Of Change Request	Number of Requirements Issues	Number of Inspections Required	Manpower Required to Make Change
107734	1933	4	3	27	7	238	12	209.3 MW

Discussion

Although we believe we have made a reasonable translation from a code oriented requirements risk analysis to a component oriented one, it is not clear that the resultant risk model would be entirely usable because no matter how we define the software entities of interest, we still do not have equal visibility of the host software and COTS. We suggest this is a fundamental problem that has not been solved by COTS advocates, particularly for safety critical systems. Next we present a framework for identifying and analyzing the cost of COTS.

Cost as the Universal COTS Metric

We focus on factors that the user should consider when deciding whether to use COTS software [SCH992]. We take the approach of using the common denominator *cost*. This is done for two reasons: first, cost is obviously of interest in making such decisions and second a single metric – cost in dollars – can be used for evaluating the pros and cons of using COTS. The reason is that various software system attributes, like acquisition cost and availability (i.e., the percentage of scheduled operating time that the system is available for use), are non-commensurate quantities. That is, we cannot relate quantitatively "a low acquisition cost" with "high availability". These units are neither additive nor multiplicative. However, if it were possible to translate availability into either a cost gain or loss for COTS software, we could operate on these metrics mathematically. Naturally, in addition to cost, the user application is key in making the decision. Thus one could develop a matrix where one dimension is *application* and the other dimension is the various *cost elements*. We show how cost elements can be identified and how cost comparisons can be made over the *life* of the software. Obviously, identifying the costs would not be easy. The user would have to do a lot of work to set up the decision matrix but once it was constructed, it would be a significant tool in the evaluation of COTS. Furthermore, even if all the required data cannot be collected, having a framework that defines software system attributes would serve as a user guide for factors to consider when making the decision about whether to use COTS software or in-house developed software. Note that host software could be developed either in-house or under contract. If the former, the in-house cost element below apply to host software.

Certainly, different applications would have varying degrees of relationships with the cost elements. For example, flight control software would have a stronger relationship with the cost of unavailability than a spreadsheet application. Conversely, the latter would have a stronger relationship with the cost of inadequacy of tool features than the former. Due to the difficulty of identifying specific COTS-related costs, our initial approach is to identify cost elements on the ordinal scale. Thus, the first version of the decision matrix would involve ordinal scale metrics (i.e., the cost of unreliability is more important for flight control software than for spreadsheet applications). As the field of COTS analysis matures and as additional data is collected about the cost of using COTS, we will be able to refine our metrics to the ratio scale (e.g., the cost of unreliability in a host system is two times that in a commercial COTS system).

The cost elements for comparing COTS software with in-house software are identified below. This list is not exhaustive; its purpose is to illustrate the approach. These elements apply whether we are comparing a system comprised of all COTS components with all in-house components or comparing only a subset of COTS components with corresponding in-house components. Explanatory comments are made where necessary. Mean values are used for some quantities in the initial framework. This is the case because it will be a challenge to collect *any* data for some applications. Therefore, the initial framework should not be overly complex. Variance and statistical distribution information could be included as enhancements if the initial framework proves successful.

Cost Elements

$C_c(j)$ = Cost of acquiring COTS software in year j.

$C_i(j)$ = Cost of developing in-house software in year j.

$U_c(j)$ = Cost of upgrading COTS software in year j.

$U_i(j)$ = Cost of upgrading in-house software in year j.

$P(j)$ = Cost of personnel who use the software system in year j. This quantity represents the value to the customer of using the software system.

$M_c(j)$ = Cost per unit time of repairing a fault in COTS software in year j. This is the cost of customer time involved in resolving a problem with the vendor.

$M_i(j)$ = Cost per unit time of repairing a fault in in-house software in year j.

$R_c(j)$ = Mean time of repairing a fault that causes a failure in COTS software in year j. This is the average time that the user spends in resolving a problem with the vendor.

$R_i(j)$ = Mean time of repairing a fault that causes a failure in in-house software in year j.

$T(j)$ = Scheduled operating time for the software system in year j.

$A_c(j)$ = Availability of software system that uses COTS software in year j.

$A_i(j)$ = Availability of software system that uses software developed in-house in year j.

These quantities are the fractions of T(j) that the software system is available for use.

$F_c(j)$ = Failure rate of COTS software in year j.

$F_i(j)$ = Failure rate of in-house software in year j.

These quantities are the number of failures per year that cause loss of productivity and availability of the software system.

In some applications, some or all of the above quantities may be known or assumed to be constant over the life of the software system. Using the above cost elements, we derive the equations for the annual costs of the two systems and the difference in these costs. In the cost difference calculations that follow, a positive quantity is favorable to in-house development and a negative quantity is favorable to COTS.

Cost of Acquiring Software

Difference in annual cost = $C_c(j) - C_i(j)$ (1)

Cost of Upgrading Software

Difference in annual cost = $U_c(j) - U_i(j)$ (2)

Cost of Software being Unavailable for Use

Annual cost of COTS software being unavailable for use = $(1-A_c(j)) * P(j)$.

Annual cost of the in-house software being unavailable for use = $(1-A_i(j)) * P(j)$.

Difference in annual cost =
$P(j) * (A_i(j) - A_c(j))$ (3)

Cost of Repairing Software

Average annual cost of repairing failed COTS software = $F_c(j) * T(j) * R_c(j) * M_c(j)$.

Average annual cost of repairing failed in-house software = $F_i(j) * T(j) * R_i(j) * M_i(j)$.

Difference in annual cost =

$T(j) * ((F_c(j) * R_c(j) * M_c(j)) - ((F_i(j) * R_i(j) * M_i(j))$ (4)

Then, TC_j, total difference in cost in year j, is the sum of (1), (2), (3), and (4). Because there is the opportunity to invest funds in alternate projects, costs in different years are not equivalent (i.e., funds available today have more value than an equal amount in the future because they could be invested today and earn a future return). Therefore, a stream of costs over the life of the software for n years must be discounted by k, the rate of return on alternate use of funds. Thus the total discounted cost differential between COTS software and in-house software is:

$$\sum_1^n TC_j/(1+k)^j$$

In this initial formulation, we have not included possible differences in functionality between the two approaches. However, a reasonable assumption is that COTS software would not be considered unless it could provide minimum functionality to satisfy user requirements. Thus, a typical decision for the user is whether it is worth the additional life cycle costs to develop an in-house software system with all the desirable attributes.

Conclusions

The decision to employ COTS on mission critical systems should not be based on development cost alone. Rather, costs should be evaluated on a total life cycle basis and RMA should be evaluated in a system context (i.e., COTS components embedded in a host system). COTS suppliers should also consider making available more detailed information regarding the behavior of their systems, and certifying that their components satisfy a specified set of behavioral properties. In addition, a formal risk assessment of requirements should be performed taking into account the characteristics of host system environments.

References

[ANS93] Recommended Practice for Software Reliability, R-013-1992, American National Standards Institute/American Institute of Aeronautics and Astronautics, 370 L'Enfant Promenade, SW, Washington, DC 20024, 1993.

[CLE97] Clemins, Archie, "IT-21: The Path to Information Superiority." CHIPS Jul 1997, http://www.chips.navy.mil/chips/archives/97_jul/file.htm, p. 1.

[JPL98] "Reusable Libraries of Formal Specifications", NASA Formal Methods web site, http://eis.jpl.nasa.gov/quality/Formal_Methods/library.html, 1998.

[KOH99] Ronald J. Kohl, "V&V of COTS Dormant Code: Challenges and Issues", Proceedings of the First Workshop on Ensuring Successful COTS Development, 21st International Conference on Software Engineering, Los Angeles, California, May 22nd, 1999, 2 pages.

[SCH97] Norman F. Schneidewind, "Reliability Modeling for Safety Critical Software", IEEE Transactions on Reliability, Vol. 46, No.1, March 1997, pp.88-98.

[SCH991] Norman F. Schneidewind and Allen P. Nikora, "Issues and Methods for Assessing COTS Reliability, Maintainability, and Availability", Proceedings of the First Workshop on Ensuring Successful COTS Development, 21st International Conference on Software Engineering, Los Angeles, California, May 22nd, 1999, 4 pages.

[SCH992] Norman F. Schneidewind, "Cost Framework for COTS Evaluation", Proceedings of COMPSAC 99, Phoenix, AZ, 27 October 1999, pp. 100-101.

[SRI98] "The PVS Specification and Verification System", SRI International Computer Science Laboratory, http://www.csl.sri.com/sri-csl-pvs.html, 1998.

[TAL98] Nancy Talbert, "The Cost of COTS", IEEE Computer, Vol. 31, No. 6, June 1998, pp. 46-52.

[VOA98] Jeffrey M. Voas, "Certifying Off-the-Shelf Software Components", IEEE Computer, Vol. 31, No. 6, June 1998, pp. 53-59.

Determining the Suitability of COTS for Mission Critical Applications

Ronald J. Kohl

AverStar, Inc.
3581 Mar Lu Ridge Road
Jefferson, MD, USA, 21755-7724
kohl@averstar.com

Abstract

Commercial Off The Shelf (COTS) products are being considered for inclusion in ever more complex and critical systems. There are known advantages and risks [1, 4, 5] for considering the use of COTS in complex systems. Yet, given the rigorous needs of Mission Critical systems or subsystems, there have begun to emerge concerns and risks about the suitability of COTS for such applications. This paper identifies some of the characteristics of Mission Critical systems (e.g. reliability, availability, correct functionality) that makes the selection process of COTS products (hardware, software, subsystems, etc) an increasingly important factor in total system lifecycle phases (design, development, acceptance, operations/maintenance and disposal). This paper presents a set of risk areas related to the use of COTS, in general, and specifically for Mission Critical systems, that would assist both the acquisition community as well as the development/integration community in determining the suitability of using COTS in such Mission Critical systems. Then, a set of risk mitigation approaches is identified; some of which have been applied to certain National Aeronautics and Space Administration (NASA) programs. Lastly, a set of steps that could lead to the establishment of a set of procedures, and perhaps even an enterprise policy on if and/or when COTS products are suitable for certain Mission Critical applications.

1 Introduction

Mission Critical System characteristics such as reliability, safety, availability, maintainability, and certification tend to have significant influence on whether or not COTS should be considered for a given application. On the other hand, COTS products traditionally have not been built for use in such Mission Critical applications. This systems needs versus intended product operational envelope poses one of the major challenges to using COTS products in such Mission Critical systems. Once the suitability of COTS has been determined, then it is possible that additional requirements may be placed on the product and/or the product's vendor prior to inclusion in such Mission Critical applications. Or it may be necessary to consider alternative products or approaches if a given vendor is unwilling to comply with Mission Critical product/system requirements. Further, it is possible that certain system requirements and expectations may need to be modified because of the inclusion of COTS products into that system. As COTS products continue to be considered as candidates for inclusion within Mission Critical systems, there will likely be additional risk factors that will be identified, and there will likely be improvements to the impacts of known risks to existing COTS risk factors. The continued pursuit and dissemination of such COTS risk factors will influence how both acquirers and suppliers decide if and/or when to use COTS products. Ongoing monitoring of this technology area, including both benefits attained and risks identified, seems to be warranted. In addition, validation of the mitigation techniques proposed in this paper is warranted, along with collecting lessons learned from projects, which may be experiencing such impacts, and those that may have identified additional mitigation techniques.

2 Background

Trends in both government and industry are to use COTS products more and more because there are recognized advantages: reduced development cost, large user base, reduced maintenance, etc. This trend seems to be increasing with no end in sight.

Yet, Mission Critical systems and applications continue to have ever more stringent and rigorous requirements for certain characteristics of the system or application. And there is every reason to expect that such Mission Critical systems will increase in number, complexity, and stringency.

Determining the suitability of any COTS products for such applications and systems requires efforts and analyses that may not be fully appreciated, understood, or implemented in many organizations. This is true of acquiring organizations as well as of supplying organizations.

Further more, there can be non-engineering pressures to use COTS products (Department of Defense's (DoD) Acquisition Reform, U.S. Government's legislation on Information Technology Management Reform Act (Clinger-Cohen), DoD's transition out of Mil-Stds to commercial standards (Perry memo), etc).

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

3 What are the differences in Mission Critical systems?

There is no agreed upon definition of Mission Critical systems. The intent is that such Mission Critical systems are more important than other systems, based on the perspective of a set of stakeholders. The problem is that more important tends to be an ill-defined characteristic. For the purposes of this paper, Mission Critical System is defined as "any system critical to success of an enterprise or a project". Mission critical systems have more rigorous and stringent requirements than less critical systems. These requirements usually have to do with quality and performance characteristics. Requirements in the area of availability, reliability, security, and safety are usually of higher priority for Mission Critical systems, and pose greater impacts on the subsystems, components, and elements of such systems. Financial systems, such as International Bank funds transfer, may have less complex functionality but the loss of availability, even for a few seconds, can have significant mission impacts to entire enterprises. Military facilities have security requirements that are critical to the mission of such secured facilities and enterprises. And human-based space programs have safety requirements that cannot be compromised.

In addition, Mission Critical systems tend to have more demanding performance requirements. It is not unusual for Mission Critical systems to have real-time, throughput, access, and response requirements that are far more difficult to satisfy and verify, especially via COTS products. Chemical processing plants have the need to monitor sensors many times per second, to ensure safety. Space propulsion systems have a need to monitor sensors and command effectors, many times per second, to correctly control launch vehicles and orbiting platforms. Security systems need to access restricted and protected databases in microseconds, and to disseminate the information from those accesses, over large networks in a matter of seconds, or less.

One last area of relevance to Mission Critical systems is the need for more stringent Verification and Validation efforts and possibly even product certification. NASA's Space Shuttle Program (SSP) requires software certification by both the developer and the independent verifier. Security systems also require product certification.

The above is intended to provide examples of Mission Critical requirements that are either unique to or more critical, none of which can be compromised, no matter what the solution's composition.

4 What are the risks of using COTS products in Mission Critical systems?

There is a growing body of information [1, 4, 5] that has identified risk areas when considering COTS products. These include functionality of the product, operational utilization, quality and reliability, maintenance costs, product volatility, and vendor viability.

These risk areas need to be assessed, and when appropriate, mitigated no matter what type of system that contains them.

However, Mission Critical places even higher demands on COTS products and vendors. Some examples are:

- It may be undesirable or even unacceptable for a COTS product to contain Dormant Code [3], the COTS product functionality for which there is no system requirement. Dormant Code can have technical, cost, schedule, and even legal ramifications that might disqualify a given COTS product.
- It may be mandatory to have insights into the product development processes to understand the likelihood of a quality product upon delivery. It may even be an acquisition requirement for all suppliers (from prime to subcontractors to vendors) to be ISO 9000 or SEI CMM Level 3.
- It may be necessary to have access to source code, in order to understand functionality and testability of a given COTS product.
- For long-lived systems, it may be necessary to have access to product information (source code, design documents, test scripts, etc) since a given version of a product may need to be operational for many years. This may require such approaches as source code escrow or third party maintenance agreements.
- Vendors may be required to produce or obtain certification of their COTS product, which often incurs legal and financial implications.

5 An overview of what should be done

The first step is to fully understand the expectations, desires and characteristics of the system or application to be developed, and to determine the priority of each of these needs. This will support the establishment of a critical shopping list as the system supplier ventures into the commercial component marketplace.

Almost in parallel, an understanding and insight into the availability and characteristics of the COTS products that could be solution candidates, needs to begin to be developed. Furthermore, insights into vendor business viability and reputation need to be captured and then monitored.

Then there must be an iterative process of requirements specification and candidate COTS solution evaluations. As systems requirements mature and as COTS product knowledge increases and improves, it will likely be necessary to revisit the matches and mismatches between requirements and COTS capabilities.

Additionally, if system acceptance requirements such as Independent Verification and Validation or Operational Certification are required, then determining the ability of COTS products, and their vendors to undergo the rigors of complying with such requirements, becomes a significant factor in the earliest phases of the system's lifecycle.

6 Specifically, what are the next steps?

1. Ensure that Mission Critical systems are not over specified. Be sure that only those components and subsystems of a given system that need to be very important are subject to the appropriate and more stringent Mission Critical requirements. As systems requirements mature and evolve, it is critical that these requirements be continually compared against COTS product capabilities.

2. Determine the capabilities of COTS products, and where appropriate the viability of their vendors. This must be performed early and often. It could be necessary to establish a commercial product market watch role to ensure that the COTS marketplace, the vendors in that marketplace, and the products produced by those vendors meet the system requirements.

3. Understand the operational profiles of the system to ensure that any operational concepts for COTS products, as envisioned by the vendor, are consistent with the operational profiles of the end system. This can be a major area for significant disconnect if not addressed early and revisited often.

4. Determine if there are additional approaches to determine the compliance of COTS products with Mission Critical requirements. Such approaches as additional testing, vendor certification, and third party product certification may be required.

5. Establish positive relationships with the COTS vendors to promote good business dealings. Such positive business relationships can ease or improve negotiations with COTS vendors, where appropriate, for access to product and process information not normally provided by such vendors (they may not necessarily say no!)

6. Understand alternative COTS products. This requires knowledge of the marketplace, the vendors and products in that marketplace and the products that are emerging into the marketplace. By knowing the full range of candidate solution components, there is reduced risk that the final solution will satisfy the full spectrum of systems requirements.

7 An example

The United State's NASA SSP recently selected a commercial GPS system (a military version of a commercial GPS system) to replace the onboard TACANs for navigation functionality. A test/acceptance program was implemented, including test flights onboard the Space Shuttle Orbiter. A respected vendor was selected from candidates and SSP began to perform a set of analyses and tests to validate the capabilities and quality of this product. In spite of what was considered the correct processes to satisfy SSP's expectations of this GPS subsystem, an on-orbit problem occurred during the first test flight on Shuttle Mission STS-91, in 1999 [2].

The nature of the problem lied in an interface between the Onboard Flight Software (FSW) and the GPS Receiver subsystem. Certain problems, not fully understood by the SSP, manifested themselves during STS-91, leading to Nav state divergence that eventually manifested itself in loss of communications between the Orbiter and the ground. As a consequence of this problem, NASA has reverted to the TACANs, has improved the interface between the FSW and the GPS subsystem (more protection), and has implemented a variety of more stringent analyses and process improvements.

What were the assumptions that were made to support the adoption of the commercial GPS receiver?

- Reduced costs to SSP.
- Leverage from military experience and testing of GPS Receiver subsystem.
- Adoption of new technology in reduced time (obsolescence was a factor).
- Intense Black Box testing would satisfy V&V requirements and expose any hidden problems.

What were the risks/problems encounter?

- Operating environment/profile was different.
- Insufficient Systems Engineering across all aspects of the GPS system, especially the firmware.
- Process rigors of SSP were not satisfied by the GPS Receiver vendor.
- Lack of insight into GPS Receiver design.
- Lack of GPS math model.
- Declining vendor knowledge on the GPS Receiver product line.

What were the lessons learned or changes made by SSP?

- COTS/MOTS should not be considered a silver bullet
- Thorough Systems Engineering, early and often, remains critically important.
- Relying on Black Box testing has limits and may be insufficient.
- Lack of insights into product designs can lead to unknown problems.
- COTS vendors should be involved early and across the lifecycle.

8 What future steps could be considered?

1. Validate the above set of practices by industry and government practitioners.
2. Contact (survey, interview, etc) current programs that have Mission Critical components and determine if they are considering COTS products. If they are, determine how they select COTS products.
3. Contact researchers (industry, government, and academia) to determine areas suitable for long term study/analysis/research.
4. Maintain an ongoing monitoring of these practices and the users of them, to reassess the validity of them and to identify new practices for consideration.

9 Conclusions

The use of COTS products in Mission Critical systems is an emerging trend, which requires sound engineering practices. Not all of these practices are fully understood or mature, yet. As the practices suggested in this paper are implemented, they will be improved and new ones will emerge. There is much to learn about effectively using COTS products, across the total system lifecycle. Moreover, there are additional risks and mitigation techniques that affect Mission Critical systems, some of which have yet to be identified. Further validation of the practices suggested here and the emergence of new practices will improve the ability of systems developers to incorporate COTS products while still satisfying the critical demands of large, complex systems.

References

[1] J. Clapp, A. Tabb, "A Management Guide to Software Maintenance in COTS-Based Systems", Mitre Corp, Mitre Paper MP 98B0000069, Nov. 1998.

[2] J. Hutchins, "Shuttle GPS Upgrade, COTS/MOTS Issues and Lessons Learned", Proceedings, ATWG Fall Conference, 1999.

[3] R. Kohl, "When Requirements are not isomorphic to COTS Functionality: "'Dormant Code' within a COTS product", Proceedings INCOSE Symposium, July, 1998.

[4] D. Reifer, T. Ragan, G.E. Kalb, "COTS Software Management: Taming the Beast".

[5] SEI, "COTS-Based Systems (CBS) Initiative", at http://www.sei.cmu.edu/cbs/index.html.

Six Facets of the Open COTS Box

(March 2000)

Daniel H. Dumas
Certified Consultant IT Architect, Network Computing
IBM Belgium
Square Victoria Regina 1
B-1210 Brussels, Belgium

Summary

Although procurement of COTS software for Defence applications has long included evaluation in terms of the products' respect for standards and norms, actual experience has often revealed shortcomings in the ability to deploy solutions based on these packages widely over a period of time. We look here at what additional factors need to be considered in order to make the use of COTS software more likely to bring continuing benefits over the life of an application system. The six aspects that are considered in the paper are:

- Presentation interfaces
- Release compatibility
- Portability
- Programming interfaces
- Security interfaces
- Management interfaces

Introduction

The advantages of using COTS software packages and components are widely known and appreciated, namely:

- Rapid availability (by definition: off the shelf)
- Lower initial costs (because fixed development costs are spread over a wider user population)
- Widespread and higher quality education offerings (again because of the wide user base)

In choosing a particular software package, an organisation will also look to such factors as the ability to use it on the platforms that are most widely used within the organisation (including possibly heterogeneous platforms), and the breadth of applicability of the solution, to understand the economies of scales and of skills that can be realised. This touches upon factors commonly referred to as the "openness" and the perenniality of the solution.

Although procurement of these offerings for Defence applications has long included evaluation of how well they respect official standards and norms, actual experience has often revealed shortcomings in the ability to deploy them widely even over a few years' time. We may wonder, then, what additional factors need to be considered in order to make the use of COTS software more likely to bring continuing benefits over the life of an application system. We will consider here six facets of the definition of "open software" that need to be taken into account in evaluating COTS offerings. These will be presented symbolically by looking at the kind of information we might hope to find written on the six sides of the cardboard box that the COTS software is delivered in.

Six Facets that should be considered in COTS software evaluation

Each of the six sides of the box that a COTS software is typically delivered in can serve to remind us of a separate, important aspect that needs to be considered in evaluating the software, in order best to ensure its long-term applicability in a particular environment.

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

Those six aspects, that we will detail in the rest of this paper, are the following:

- Presentation interfaces
- Release compatibility
- Portability
- Programming interfaces
- Security interfaces
- Management interfaces

Presentation interfaces

Let's consider the top of the box first - and with it, the first thing that you see when you look into a new software: the presentation interfaces.

This aspect , especially when we are dealing with graphical interfaces, has an undeniable emotional impact at the time of product selection. Ergonomy of use in some cases may merit a detailed and objective study, including how long it takes to accomplish some benchmark series of tasks, for both the inexperienced and the experienced user. It remains, of course, difficult to quantify to what degree evaluations are conditioned by more subjective elements such as the use of colour and graphical elements, and the aesthetic elements of information layout.

The nature (and in particular, the complexity) of the presentation interface can also have an impact on performance in environments with bandwidth or processor constraints. It is appropriate that this be taken into consideration at the time of evaluation, but this aspect is rarely apparent in a demo or pilot environment, where bandwith and processor capacity is typically unconstrained.

But beyond considerations such as usability and attractiveness, the choice of presentation interfaces has a real impact on the ability for a product to be used at various locations within the organisation, and to integrate with various other applications.

If a single type of user interface is used, an approach that presents significant advantages is to communicate to the user interface with a Web Browser-supported data stream, such as HTML, XML or client-side Java functionality. The Web Browser, of course, has the distinct advantage of running on multiple platforms.. It can also connect to multiple servers simultaneously. This allows easy passive integration, as well as active integration via hyperlinks. Increasingly, as well, it can be used to provide slightly different presentation interfaces to users according to their individual preferences, through the use of style sheets.

An approach that goes further than this is to support multiple user interfaces. Indeed, if sufficient consideration is given, at the time of application development, to the separation between presentation functions and business logic, the same application can be designed to work indifferently with various interfaces. As possible interfaces, we might imagine the following:

- A non-graphical, or transactional, interface to the server functions, via a specific Application Programming Interface or a Messaging interface
- A specific Client GUI (graphical user interface)
- A standard Web Browser GUI
- An interface to portable devices such as Personal Data Assistants, cell phones or pagers
- An output interface such as print, e-mail, pager or FAX
- A telephony user interface, which could be touch-tone or voice activated

There are multiple approaches possible to providing universal presentation interfaces. The intelligence required in order to adapt the presentation format to a particular device can either be located in an intermediate server, or *transcoder,* or can be built into the device itself, based on standard datastreams that are defined to be applicable for data transmissions to a wide variety of devices. It is important to see which of these approaches is adopted by a particular software package, and to evaluate how the approach fits in with the planned deployment of various user interfaces within the organisation.

The idea of universal access to applications is gaining credibility through recent advances in the standardisation of separation between content description and layout, based on the use of the eXtensible Markup Language (XML - which, in contrast to HTML, is not a markup language in the sense of presentation layout, but a content description language) and the eXtensible Style Language (XSL) specifications.

Both the transcoding and the device-resident style sheet approach can provide increased flexibility for customisation of the presentation interface. Typically, though, they will be used for different types of applications.

The approach that will leave the greatest flexibility for customisation of the look of the presentation interface by individuals will be the approach based on a style sheet selected at the device. This will generally allow the user to adjust the presentation interface without requiring any modification to the business logic. It's an approach that has advantages for the editor of the software as well: it allows them to avoid developing specific customisation Application Programming Interfaces (API's), and will reduce needs to provide access to source code, with the accompanying negative impacts that has on maintainability of code and on the ability to protect intellectual capital and software assets.

There are other types of applications where the transcoder approach is more powerful, of course: in cases where the datastreams are non-standard or unstructured, for instance. One application where use of this kind of service has appeared recently is in providing multilingual interfaces to a single-language application. The transcoder in this case is used to accomplish translation of text on-the-fly.

Release compatibility

Let's go back to the imaginary box that our software has just been delivered in, and take a look at the front side now. What kind of things might we find there? Version X.Y.Z. New! Improved! Bonus!: now includes product ABC (demo version).

Questions we might ask upon seeing all this include:

- Why the new release? (as well as: when was the previous release? when is the next one planned?)
- What statement of requirements motivated the new function and improvements?
- How does the additional product included in the package affect me and what dependencies does it create?

COTS software is fundamentally oriented to a mass market. New releases can serve a number of purposes in this environment:

- They are a delivery mechanism for error maintenance
- They are a mechanism to generate renewed interest
- They incorporate new technology
- They allow adjustments in positioning relative to competitive products, and to products the vendor owns, has acquired, or is forming marketing alliances with

Because market share is an important consideration, COTS products typically try to cover as large a spectrum of function as possible - sometimes earning them the reputation of "bloatware"! This inevitably results in their containing features that are not strictly required for a particular application. Additionally, and more importantly, the focus very often turns to rapid product cycles rather than to managed change.

In this context, it is not infrequent to see problem determination and the application of fixes reduced to very minimalist proportions: install the next release and hope your problem goes away. The policy of maintenance for releases for a particular COTS product certainly merits investigation. Is corrective maintenance available between releases? How can it be delivered?

Release “churn” has a negative influence on the length of availability and on the effective support period of a release. How frequently do releases change? Is this acceptable in terms of the period foreseen to roll out a product to the various users in the organisation?

Change management considerations apply not only to the area of corrective maintenance, but to the implementation of functions that have been requested as part of the product requirements process. Historically, input to the requirements process has been restricted to a small number of influential players. The advent of the Internet is changing that in certain IBM and Lotus laboratories, where requirements from a much broader public of developers is solicited during the product development cycle.

Eventually, even if hopefully not during the initial rollout period, the organisation will probably end up considering migration of users from one release to another, or one version to another, of any given COTS product. A number of other important questions will inevitably arise at that time: will the things that I have customised or developed continue to work with the new release (forward compatibility)? Will the things that I was doing with the previous release continue to work with both the new and the old release if I perform them with the new release (migration compatibility)? Will the fact of using the new functions in the new release prevent me from interoperating with users using the old release (backward compatibility)? Is there some way to configure so that compatibility is ensured (e.g., disabling the use of the new functions or new data formats until migration is complete)? What needs to be done to move users from the old release to the new release, and data from the old release to the new release (migration planning)?

Although forward compatibility is widely practised, and backward compatibility is sometimes possible, the ability to configure for automatic backward compatibility is rarely foreseen. These last considerations can however be especially important in situations where upgrade decisions are taken by independent entities or distributed entities that need to interoperate.

Obviously, these are all essential change management considerations that need to be understood before a given organisation leaps into a new release. But when we consider the question of release compatibility across organisations, the question gains a new dimension of complexity: Organisations throughout the world are not marching in lockstep. Different organisations are doing different things at different times. No version plan could ever be made that would make all of a COTS supplier's customers happy.

Customers have to have the discipline to navigate through releases, and have some restraint to do version control. Industry on the other hand, who too rarely make public commitments to maintenance of a particular release, could do better to provide maintenance of prior versions over a fixed minimum number of years. But this does not appear to be a prevalent trend. I do have one COTS software box that states: "Maintenance will be provided until .... No maintenance will be provided after that date". It just happens to be for IBM DOS 4.0!

New releases can also entail additional licensing fees in addition to the unaccounted human costs associated with the installation, configuration and problem determination efforts required for those new releases.

Portability

Moving on to the right side of the box, we might see a text such as the following: “Requirements: Windows 95 or 98, Intel Pentium 133MHZ or greater with a minimum of 24MB, a Sound Blaster compatible sound card and SVGA graphics capability configured for at least 800X600 resolution." A number of questions might typically come to mind:

- Will this software be applicable to my other machines that work with other hardware and/or other operating systems? Or will I at least be able find the equivalent software available for the other operating systems?

- Will this work with the new operating system version (NT 4.0, Windows 2000, etc.) that I am planning on installing (or that I will be forced to install for some other reason)?
- Will it interoperate with my other systems, or with the systems in other organisations that I need to deal with?
- How scalable is the package? Can it take advantage of additional memory, additional processors, additional machines or more powerful machines, in order to accomplish more work?

Portability has to do with flexibility across technologies and over time. Whereas in the considerations concerning release compatibility we were considering constant platforms and varying software, here we are considering constant software and varying platforms.

We are looking to be able to deploy a COTS-based solution widely and to keep it viable over a number of years, eventually in a number of different organisations that need to work together. In order to accomplish that , we need to accommodate changes in technology and possible changes in hardware and operating system vendor strategy. The rhythm of change of hardware and operating system technology continues "relentlessly" as well! It is therefore desirable for the software to have a high level of abstraction from the hardware and operating system level.

Packages and components that we can characterise in this way generally are designed to run on multiple platforms today. The greater the number of platforms supported, the greater the effective openness of the software.

A major advance in portability has occurred recently with the advent of the Java Virtual Machine (JVM) and the standards that have been defined in the area of application development based on the Java language. The JVM, implemented across various hardware and operating system platforms, allows the same "100% Java" byte codes to be executed in the same way regardless of the instruction set and services of the underlying physical machine and operating system. The principle is: "Write once, run everywhere". Though there is a certain overhead associated with this additional level of abstraction, techniques such as Java compilers and Just-in-time Java compilation now allow performance-critical processes to achieve results that reasonably approach the performance of native instruction-set execution for equivalent functions. Java-based applications that correspond to user performance expectations are increasingly becoming available, and this trend can be expected to continue

Portability is also affected by the architectural approach followed by the solution. Functions that risk being dependent on specific platforms can be separated from the other functions, and accessed via a protocol that allows the function to be located elsewhere, in order to make the overall functions accessible from a wider range of platforms. This is essentially acknowledging that portability is most important across the machines that have the greatest number of instances installed, while an organisation can afford to have a limited number of servers for which portability is not considered an issue (typically, which have specific characteristics of availability, performance, security, or other criteria).

In terms of the communications protocols used between the dissociated functional layers, we can speak of synchronous protocols (requiring both sender and receiver to be active and accessible - e.g. as in the use of a TCP socket-to-socket protocol) and asynchronous protocols (allowing processing to go on with guaranteed delivery at a later time, when the receiving application is not active or accessible, e.g. as in MQSeries message queuing). Both of these types of communication are available on a wide variety of platforms.

The various approaches to separation of functional elements can be characterised in terms of architectural tiers. Although there are different approaches to counting the number of tiers involved, the following should be generally recognisable to everyone, at least in theoretical terms. It is presented here in an order that corresponds in general to an increasing order of portability:

- Monolithic applications
- Two-tier client/server applications (presentation function located in the client, communicating to business logic in an application server with integrated data store)
- Three-tier client/server applications (presentation function located in the client, communicating to business logic in an application server, communicating in turn to a data store server)
- Four-tier client/server applications (simplified presentation function located in the client ("light client" = browser), some presentation logic located in the web server, business logic in an application server, communicating in turn to a data store server)

Separation of the function into additional tiers increases their independence. It makes it easier to "live with", and integrate to, those isolated elements in the overall solution that are the most difficult to change and that may have the greatest need for stability and the least portability.

Programming interfaces

On the back of the box, we often find some information about the interfaces supported - though perhaps not nearly in the detail needed in order to put them into practice! In order to extend a package or integrate it into a larger context, programming is often necessary. How easy - or difficult- will that be with the package at hand?

Often this is a question of experience, and Internet sites for developers and user forums can provide interesting insights sometimes. But there are certain interfaces that provide very high-level function, and therefore can be used very productively. An example of such high-level function is that provided by such a specification as Enterprise Java Beans (EJB). In addition to the support for the Java language and Object Request Broker for connection to the function of other (eventually remote) objects, the EJB container provides transactional functions (such as management of the unit of work and scope of recovery), session management functions (via EJB Session Beans), persistent data store functions (via EJB Entity Beans) and access control functions.

We can distinguish multiple levels of openness in the area of programming interfaces. We can find:

- Undocumented/unofficial interfaces (true "proprietary interfaces")
- Official interfaces with limited programming function (e.g. "wizards")
- Official interfaces in a proprietary programming language (e.g. Oracle PL/SQL script)
- Official interfaces in a non-proprietary programming language (e.g. the use of COBOL, C or Java)

The usefulness of the limited-function interfaces is also conditioned by which middleware they foresee. Sometimes a small door can open onto a very large playing field! Take, for instance, a communications interface to SMTP, to EDI, or to MQSeries. Or take an SQL API, allowing the relational database to serve as an integration point to other processes, which might run asynchronously, or synchronously through triggers or through stored procedures.

The web browser, with its capability of being in fact a client to multiple servers at the same time, even on the same web page, and with its programming capability, also provides an integration point for applications, provided of course that the application foresees using a web browser interface.

For server-type implementations, additional technical analysis of the limitations of the interfaces can also be important. Such items as their support of caching, buffer handling, threading and connection pooling can have an important impact on their scalability..

Security interfaces

On the left side of the box, it would be nice to see something about how the package handles security and access control. Does it provide and use its own system? Does it build on the facilities of the operating system?

Even the most mundane applications (for instance - a word processor!) may have to handle personal, restricted or confidential information. Issues such as user identification, authentication, access control, encryption and non-repudiation must be addressed. At times, the operating system can be counted on to deliver these functions (when, for instance, it is a question of providing access control for information located on the machine). At other times, certain aspects need to be handled on an application level (for instance for data that needs to be transported, that needs to be digitally signed, that needs additional granularity of access within a given file, etc.).

The interfaces available within a COTS package can determine whether these aspects will require (or even allow) specific development, whether it will work with existing infrastructure (such as smart cards, readers, digital certificates, directories, existing definitions of users, groups and roles, encryption algorithms, etc.) or whether a separate infrastructure will need to be set up and maintained.

One approach taken in this area is that of providing a standardised interface to an external "pluggable security module", which can provide cryptographic services of various sorts. This is the CDSA model, originated by Intel, and used in various recent IBM products today. It is also the model being used by Lotus to separate the grade of security provided in a particular environment from the actual standard function of the underlying messaging product.

Management interfaces

And now for the side that everyone forgets to look at. Until there are problems, that is!

Typically we are going to be rolling out COTS software to large numbers of users, perhaps in various distributed locations, and then we are going to have to maintain an inventory of who has what level, detect problems that might occur, manage the application of maintenance, perform backups, provide for recovery, maybe provide remote debugging or remote assistance, operate; monitor performance and availability, etc..

How do we manage the cost of doing that?

The Management and Monitoring Interfaces provided by software applications can in theory support multiple objectives in the organisation, including failure detection, deployment tracking for initial roll-out or maintenance, detection of misuse, assembling and tracking performance data, remote operations, assistance or debugging, etc. But there must also be some coherence in the interfaces provided across the various applications in order for this to be viable.

COTS software will generally lack these capabilities to manage themselves. What is more important is that they interface to some central management and monitoring facility that does have these capabilities. Here is an area where the Programming APIs can come to the rescue. They can allow alerts to be implemented relatively easily, based on some reusable standard functions. An example of this are the Java classes (functions) provided to send alerts to the Tivoli Enterprise Management facilities.

Conclusion

By considering these various facets of openness, IT architects can improve the use of COTS components in complex Information Technology projects. It took some time for software companies to embrace the open movement. Today, with companies increasingly responsive to customer needs, and new technologies addressing a broader range of interfaces, we are ready to move forward to a more comprehensive definition of openness, and, as shown by a few of the examples from IBM related in this paper, we can expect that the companies providing Commercial Off-The-Shelf software will be prepared to respond.

Lotus White Paper on COTS for Military Crisis Applications

Patrick Fournery
Director of Technology
Uffe Sorensen
Director of Business & Strategies
Lotus Development S.A.
Immeuble Lotus, La Défense 6
35-41, rue du Capitaine Guynemer
92925 Paris La Défense Cedex
France

Summary: As businesses evolve to e-businesses, it is interesting to observe how the civilian requirements related to COTS software increasingly resemble the military crisis-mode requirements in terms of continuous operation (100% availability), vast scalability (Internet community), absolute reliability (transactional integrity), total security (numerous "enemies" with malicious intent in a 1B user wired community), flexible and manageable interoperability (alliances, mergers and acquisitions must be almost instantaneous and fully controlled). As COTS software vendors satisfy these civilian requirements, it will eventually facilitate military use. Inadequate software will naturally be supplanted in the marketplace by capable technologies.

"The central event of the twentieth century is the overthrow of matter. In technology, economics and the politics of nations, wealth in the form of physical resources is steadily declining in value and significance. The powers of mind are everywhere ascendant over the brute force of things. .... the most powerful corporation is .... the one with the ability to rapidly turn ideas and thinking into new products, new services and new business ! ....
In this environment, SW is not just product. It is a competitive weapon !"
Microcosm by George Gilder

Introduction

This paper is intended as an introduction to the thought-process in Lotus around Commercial Off-The-Shelf software in the context of military use.

We think that the civilian requirements placed on the software vendors actually more and more resemble demands formerly only raised in military contexts. Clearly, certain aspects of security and managing sovereignty will need specific adoption to military purposes. I deliberately write "adoption" as the fundamental COTS software is unchanged with appropriate additions and modifications for military use. A good example of this is the Lotus Domino Defense Messaging – the fundamental messaging capabilities enhanced with military grade capabilities.

In this paper we distinguish between the use of COTS software in crisis and non-crisis mode for administrative, planning, operational and conflict applications:

Administrative Applications

The use of COTS software in applications operated by mainly non-military staff dealing with non-crisis issues has clear advantages over RYO as recognized by almost all non-military organizations:

- Lower TCO – purchase price, limited education requirements around the End User Interface as most evolve towards similar paradigms, limited integration costs as standard interfaces are observed, lower maintenance cost.
- Faster implementation for quicker problem resolution.

These non-crisis mode applications are most adequately addressed by COTS software and will not be further elaborated here.

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

Planning Applications

Material planning and similar applications have two aspects - peace-time operation where resource planning is no different in military as compared to civilian organizations, and war-time operation where resource planning becomes highly critical. However, the same software must be used in both situations to ensure proper experience by civilians and military staff alike. Thus, requirements for this software is as for conflict applications.

Operational Applications

Peace-time operational applications, like surveillance, tracking, etc, exhibit exactly the same requirements from a military perspective as conflict applications - and indeed are also needed during war-time. Thus, requirements for this software is as for conflict applications.

Conflict Applications

RYO software has been the mainstay of these applications, although standard operating systems (UNIX derivatives, Windows NT) are increasingly the platform. The requirements which must be met by software in this category are further discussed in the following paragraphs in terms of Scalability, Availability, Reliability, Security and Interoperability.

Scalability

As civilian applications are increasingly made available to a World Wide community via the Internet - or just inside a growing enterprise conglomerate - COTS software for applications such as electronic mail, discussion databases, electronic publishing, document management, workflow, etc, must provide dramatic scalability.

Both in terms of server technology infrastructure supporting "few" to "millions" of users as well as an ability to deploy similar numbers of clients. Both when specific client side software is needed and when standardized clients (Browsers, IMAP4, etc. clients adhering to the appropriate Internet standards) are used.

Not only must a large number of users be sustained continually, however, ability to handle peek-load situations become critical (anecdotal examples are government releases of key white papers or investigatory reports creating massive peeks in traffic to a web-site).

COTS software with inadequate scalability will be supplanted in the marketplace by capable technology as the applications are being recognized as mission critical to most organizations.

From a military viewpoint, similar scalability requirements exist - both in terms of sustained load, but also in terms of concentrated periods of extreme activity. The similarity in requirements doesn't obviate the need for rigorous testing of infrastructure capacity and robustness in the military scenarios, however, the military application will not raise scalability requirements beyond for civilian use.

Availability

Outages of any particular software solution originates from one of several situations:

- Unplanned outages resulting from failure in the operational environment: These situations go beyond the particular software package, however, places requirements on the deployed operating environments and on the ability of the server software to support capabilities such as clustering and fail-over to minimize impact.
- Planned operational down-time for maintenance or other activities. Again, duplication through a clustering solution can provide increased / 100% availability.
- Unplanned outages resulting from software defects: The total impact here is directly related to the ability of the software author / provider to establish a work-around or providing a fix.

For the non-defect situations, the same Darwinistic viewpoints as expounded elsewhere leads to appropriate technologies surviving in the marketplace and being omnipresent.

For defect situations, identification and management of software defects should be understood: It is highly likely that de-facto testing of certain COTS software go beyond actual testing carried out with military RYO software due to sheer vastness of civilian deployment. Ultimately, in a critical defect situation the military becomes dependent on one of two:

1. Access to the COTS software vendor - and the ability / willingness of the vendor to provide a fix / workaround.
2. Access to the internal RYO software developer - and the ability / willingness of the IT department to provide a fix.

No clear prioritization can be done on either of these undesirable situations. However, what should also be discussed here, is the ability to even update the defunct software in a distributed network during a military conflict and other factors influencing the total availability of a particular solution.

Security

Involves two fundamental aspects: Confidentiality and authentication. Both are achieved through cryptography (symmetric and asymmetric) based on confidential and/or private/public paired keys, where the key length is the basic differentiator between civilian and military use.

Historically, the US DoD embargoed exportation of strong encryption technology outside the US, however, this trend is changing and strong encryption is being deployed for several key Internet business applications (most notably financial).

A key example to explore here is secure messaging:

- From a civilian viewpoint, messaging has evolved to require secure messages, which have been defined by the Internet Engineering Task Force (IETF) in the Secure MIME standard, allowing transmission of secure messages within the Internet, independently of the originator and cryptographic devices used on the transmitting and receiving side.
- From a military viewpoint, STANAG 4406 defines the NATO military community's protocols and standards for achieving interoperability amongst member nations. Most notably, STANAG 4406 embraces Secure MIME v3 for secure messages [v3 pending ratification by the IETF].

Lotus has been a leading provider of secure messaging solutions – signing and encrypting messages are natural features to Lotus Notes and Domino electronic mail users. A natural part of the evolution of Lotus secure mail was the complete adoption of the S/MIME v2 standard in the latest product releases.

As the provider of the most pervasive and secure messaging platform, it was natural for Lotus to also explore military messaging: Lotus introduced a special Defense Messaging Solution in the US as a natural extension to the commercial versions of Lotus Notes and Domino software.

A technologically similar solution was developed for the European market and made available as the LDDM (Lotus Domino Defense Messaging) solution with military grade security independent of the US DoD.

An essential feature in LDDM – which is also recognized as a key commercial requirement – is the ability to support country specific demands for sovereignty. Lotus provide the ability in the military augmented versions of the software to provide a owner defined and managed encryption scheme.

Generally, COTS software is evolving towards "plug-able" encryption modules – not only for specific solutions as LDDM, but generically for any application where sovereignty is essential, which today embraces almost all e-business web applications.

Reliability

With the rapid trend towards use of COTS software in e-business deployments and having mission critical transactions originate on the Internet, customers are forcing COTS software vendors to ensure transactional integrity: Civilian and military requirements towards having mission critical transactions executed once and only once are identical - and appropriately handled in COTS software today.

Interoperability

COTS software is by customer demand converging on a set of open standards, most notably Internet related standards. The effect is not only vendor independence and ability to integrate diverse applications with limited effort, but also the enablement of individual operational units to connect and interact on demand maintaining full control within each unit over the external factors that can influence the unit.

Customer demand is also forcing COTS software servers to expose their services to a common, robust programming model (most notably CORBA), which limits the investment needed in skills to utilize the services for specific tailored applications.

Conclusions

Although the specific intents of software applications for civilian versus military use are very dissimilar, the overall requirements are converging as a result of the increasing role of the Internet in connecting all businesses and consumers. We remain convinced that Lotus' COTS software with appropriate military amendments can play a significant role in military crisis applications.

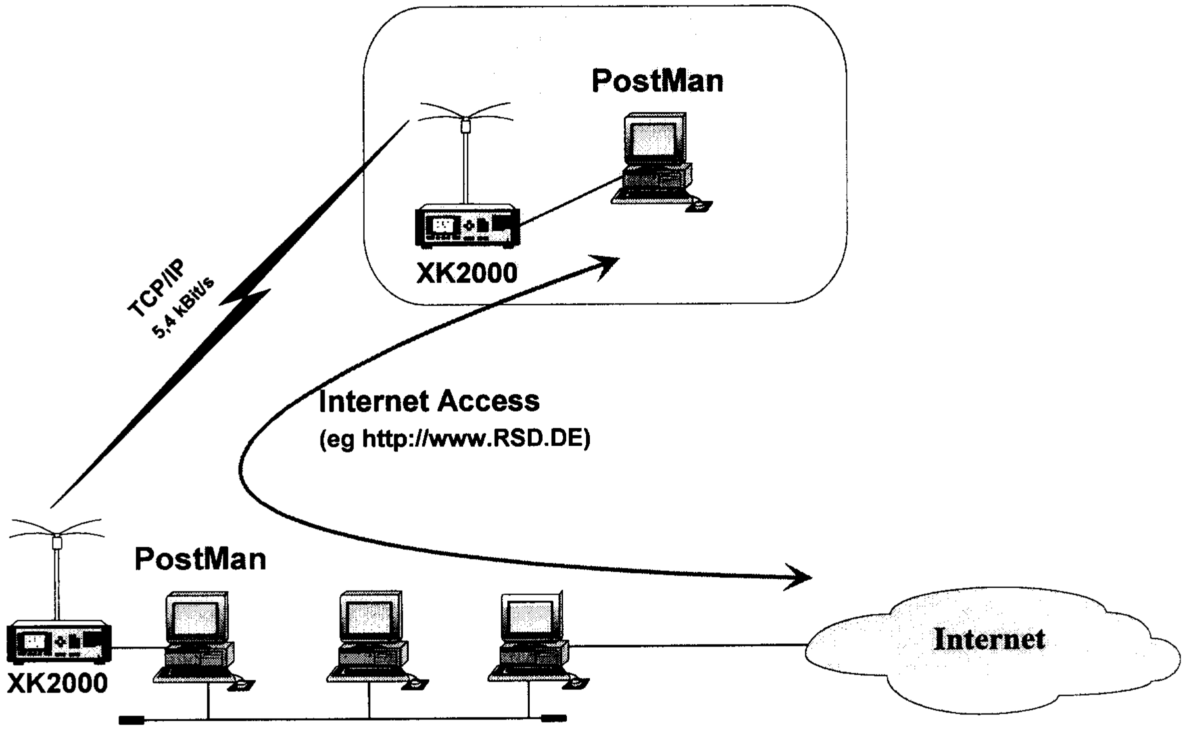
Wireless TCP/IP and Combination with Broadband Media

Thomas A. Kneidel
ROHDE & SCHWARZ GmbH & Co.KG
Abt. 2WFM
Mühldorfstrasse 15
81671 München, Germany

Objectives:
The presentation shows products for new applications (mobile IP) by using cots hardware and software components. This cots-components are implemented and adapted to fulfil services in the commercial and military field.
The following part describes the technique more in detail.

Up to now the demands of the military command were implemented in special – mostly analog – communication networks. These, however, present the great disadvantage that they are not interoperable or only to a limited extent due to the different proprietary protocols used. Among all these protocols the **TCP/IP protocol** is evolving as the **international standard** for data exchange across network borders. The TCP/IP protocol used worldwide on Internet or in X.400 networks guarantees interoperability on different computer platforms irrespective of manufacturer and operating system.

Rohde & Schwarz developed a software solution, called *PostMan* that enables **transparent implementation of the TCP/IP protocol** at the HF air interface and so ensures unhindered transmission from wire to radio communication networks. For the first time in history, for example, E-mails to or from any Internet address can be sent or received from a ship across thousands of kilometres. Even Internet surfing via short-wave with commercially available browsers is possible for every mobile station. Every TCP/IP-based application can be carried out via radio using *PostMan* which covers the entire HF/VHF/UHF band.



Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

Fig. 1: Wireless TCP/IP via HF

In some regionally limited radio networks TCP/IP-based communications are already being implemented. PostMan allows these networks to be interconnected even across large distances to obtain a full-coverage network. Within the framework of a trial at the material inspection agency of the US Army Communications Electronics Command (CECOM) this **interoperability** was tested using existing VHF/UHF data radio networks. Separately operating radio networks of the SINCGARS (Single Channel Ground and Airborne Radio System) and EPLRS (Enhanced Position Location Reporting System) type were interconnected by means of PostMan and data were exchanged across different radio links without any interaction.PostMan not only links the individual radio networks but also enables access to wire communication networks as for example Secure Intranet (SIPRNET).

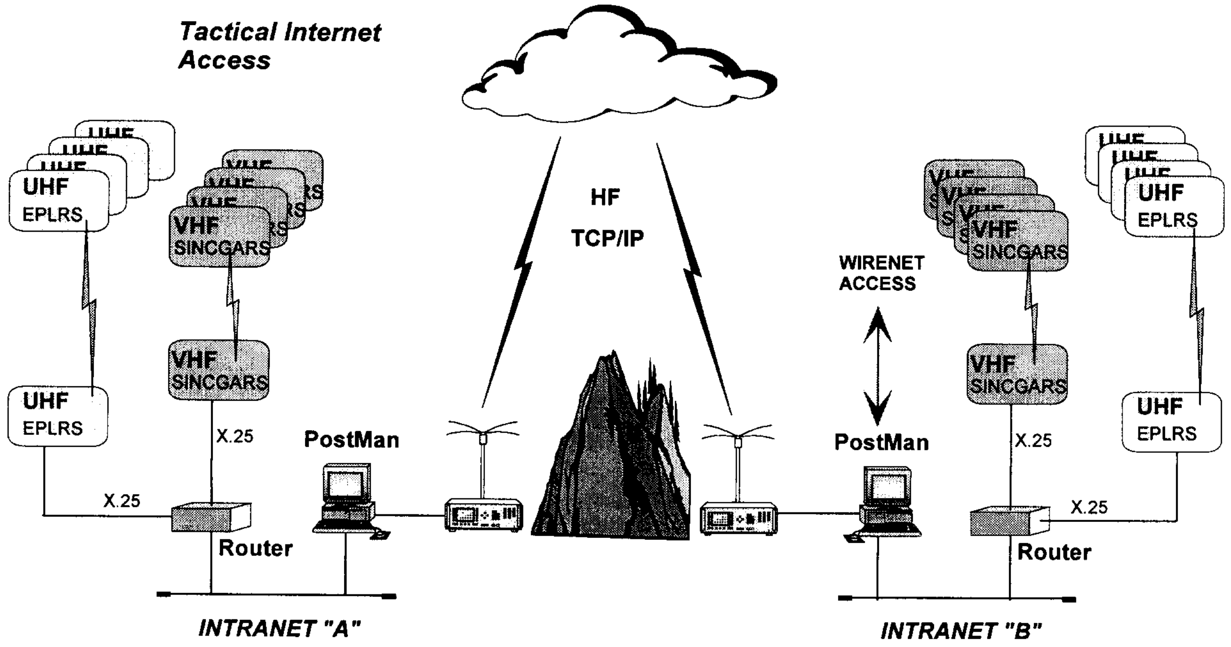


Fig. 2: CECOM Trail

In addition to transparent TCP/IP access via radio, the software package *PostMan* contains an intelligent E-mail system. This **E-mail client** is especially adapted to the requirements of a radio-based communication network (based on message application programable interface – MAPI – of MS Exchange).
In this connection, the E-mail-client contains a new **address format RSPeer** that ensures the direct delivery of the message to the computer of the addressee. The message is physically available on the hard disk of the recipient, the usual detour to the central post office being avoided. This delivery procedure excludes any misuse of and unauthorized access to the mail traffic of a network. Moreover this format ensures that one's own information is secure. This type of addressing also minimizes the data exchange on the frequencies available and so eases the traffic load of the radio network.

The messages exchanged are furthermore protected by integrated **encryption** with an algorithm that is stored on a PCMCIA card.

PostMan allows structures and **network configurations** to be defined as required. Traditional hierarchical **official channels** can be implemented. If requested, the horizontal distribution of E-mails, which is often regarded as a disadvantage in military applications, can be suppressed.

In addition to HF/VHF/UHF radio, various other transmission media such as SatCom, ISDN or GSM may be used. *PostMan* optimizes the utilization of the available media by **alternative routing**. Should the medium intended for information transmission be interrupted, *PostMan* dynamically and automatically selects an

alternative medium (according to a priority list) and continues transmission. Prior to selecting another medium *PostMan* checks whether the addressed station can be reached otherwise, eg via a relay station (**alternative paths**). The automatic change to alternative transmission media is a special feature of *PostMan* which no other E-mail system offers.

The E-mail client of *PostMan* possesses all the functions a modern E-mail system has to provide. This includes logging of all actions in a log book and assignment of different priorities to messages and addressees. Additional transmission acknowledgement and individual preselection of the time at which the message is to be transmitted support the use of *PostMan* in radio networks.

The *PostMan* software package from Rohde & Schwarz gives radio networks access to the existing worldwide wire communication networks and their applications. The E-mail client moreover optimizes the utilization of electronic messages in military applications.

Combination with broadband media

The ever increasing volume of information in the data networks congests the available transmission channels. Modern broadband transmission media point the way out of this bottleneck. Especially in conjunction with conventional narrowband media, the new media ensure efficient utilization of the channels and achieve so far unattained data rates.

For stationary applications, the data highway to the office or living room can take on a variety of forms. Wireless local loops can be implemented via the air interface using microwave links or the access to data can be accelerated by means of fiber optics, power lines or TV cables. Mobile applications face serious limitations with the data transmission bottleneck and the low data rates being at the root of the problem. Modern technologies (eg ADSL), intelligent management systems, complex coding methods and adapted protocols (eg WAP) open up new approaches. Same as in the case of stationary applications, considerable improvements are expected of the use of new broadband transmission media. At present, the role that third-generation mobile radio with UMTS (universal mobile telecommunication system) is going to play in this scenario is not yet clear. The following media look promising in easing the situation:

- DAB - Digital Audio Broadcast
- DVB-T - Digital Video Broadcast Terrestrial
- GBS - Global Broadcast Service
- GSS - Global Satellite Service

The new media are suitable for both broadcast applications and point-to-point connections.

Broadcast

In broadcast operation, individual subscribers or entire groups receive data from a center without having to have access to the source of information. Internet contents such as newspapers or market reports are emitted to subscribers via Internet push services. The subscribers store the received data in their PCs. Using standard web browsers the data in the PC can be read at any time. This fast and unidirectional way of distributing information is of great advantage in military applications since the recipient of the message does not have to send an acknowledgement which could enable the enemy to intercept and locate.

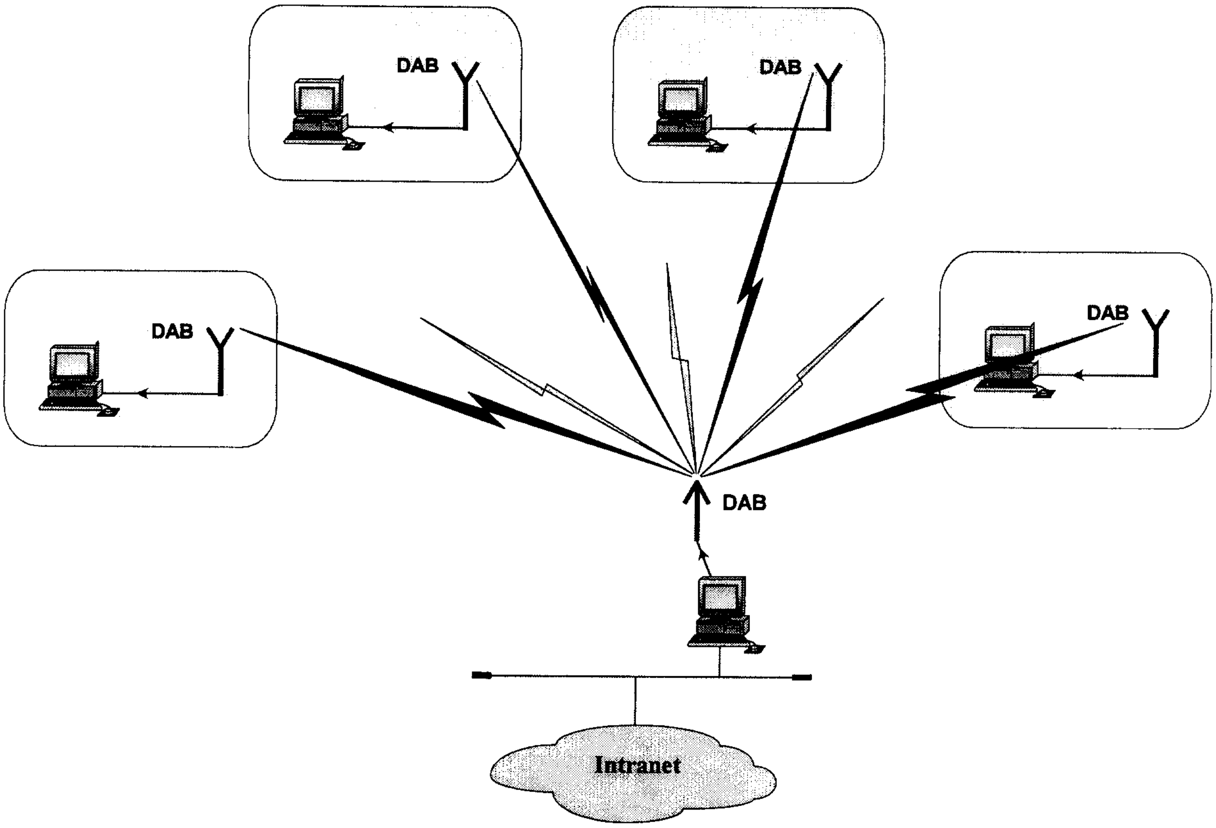


Fig. 3: Broadcast operation via DAB transmitter

Point-to-point connections

With point-to-point connections, the desired information can be called from an interactive data network, provided that the user is connected to the information source. Broadband media however have the disadvantage that they normally require considerable technical outlay (weight, space, power supply, ...) at the transmitter end. Moreover they are often only partly suited for mobile applications. A solution that can easily be implemented on an aircraft carrier, for example, may well be literally unbearable for an infantryman. The fact that most interactive multimedia applications are however based on asymmetrical communication involving short queries and extensive replies favours hybrid solutions. The access to the Internet, for example, can be implemented via narrowband media such as shortwave, tactical radio, TETRA or GSM, while the data themselves, which are usually quite voluminous, are returned via broadband media. This ensures an extremely efficient utilization of the available resources and at the same time makes a virtue of necessity.

The choice of the transmission media is essentially determined by the distances to be covered. Additional decision criteria are data volume, transmission speed and security.

LOS range

DAB is regarded as the most promising broadband media for the LOS (line of sight) range. DAB was originally designed for the transmission of sound to mobile and portable receivers, but it is also an ideal platform for the secure transmission of digital data of any kind. DAB networks operate as single frequency networks (SFN) and so ensure frequency economy. Information is distributed from various transmitters in program multiplex to different receivers at the same frequency. OFDM (orthogonal frequency division multiplexing) coding as the modulation method provides excellent transmission quality, which ensures reception even at high speed (up to 900 km/h).

The data are transmitted in band III and L band (174 MHz to 227 MHz/ ≈ 1.5 GHz) at rates of up to 1.5 Mbps (megabit per second). The information in IP (Internet protocol) format is inserted into the ETI (ensemble transport interface) data stream in line with ETSI-ES-201-735 (European Telecommunications Standards Institute).

The DAB counterpart for the transmission of TV signals is DVB-T that has similar characteristics and capabilities as DVB but enables mobile reception at low speed only.

To distribute information, for example in the deployment of rapid crisis reaction forces, mobile DAB transmitters can be set up without any problems even in remote areas.

As regards point-to-point connections in military applications, the narrowband back channel is preferably by means of VHF/UHF radio links. Basically, GSM (global system for mobile communications) could also be used for this purpose. GSM however requires an intact infrastructure that cannot be guaranteed in military scenarios. Combat net radios on the contrary set up reliable links even under conditions of jamming and impede interception.

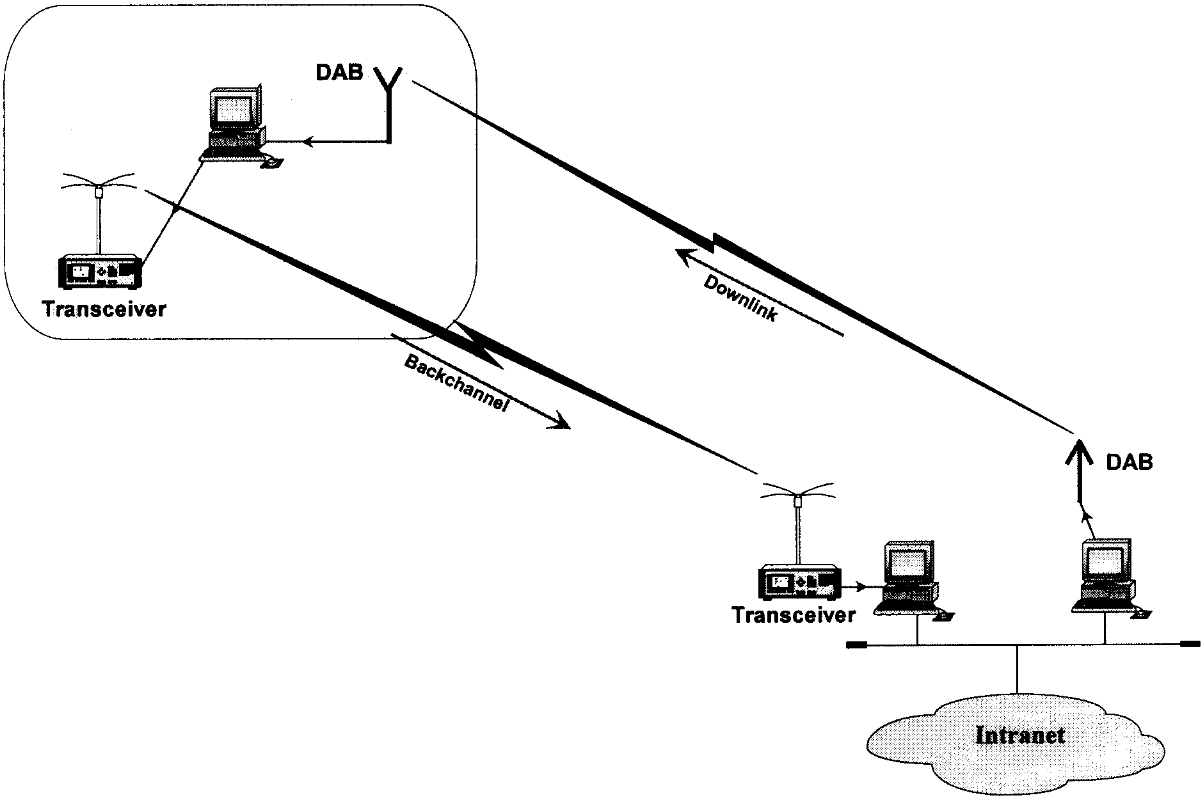


Fig. 4: The Rohde & Schwarz product range includes all components necessary for point-to-point operation: tactical M3TR radios, DAB and DVB transmitters and IT product PostMan.

The mix of DAB and tactical radio opens up a wide variety of applications. If a battalion command post has to be relocated for example, database updates can be transmitted to the new site within seconds. Awkward troop movements required at present for database update would so become a thing of the past.

Supraregional sector
Another already frequently used method is the integration of digital TV into the world of communications. With this approach, the desired information is requested from the Internet via the usual transmission paths. The reply data stream, however, is routed from the source (server) to the operation center of the satellite network and transmitted to the user via a fast, broadband satellite downlink.
This technique of course permits the pinpointed transmission of information in broadcast mode to individual users or groups.

It is basically intended for the wide-base and consumer market, but can also be employed in mobile radiocommunications. Data are called via HF for example and returned via satellite.

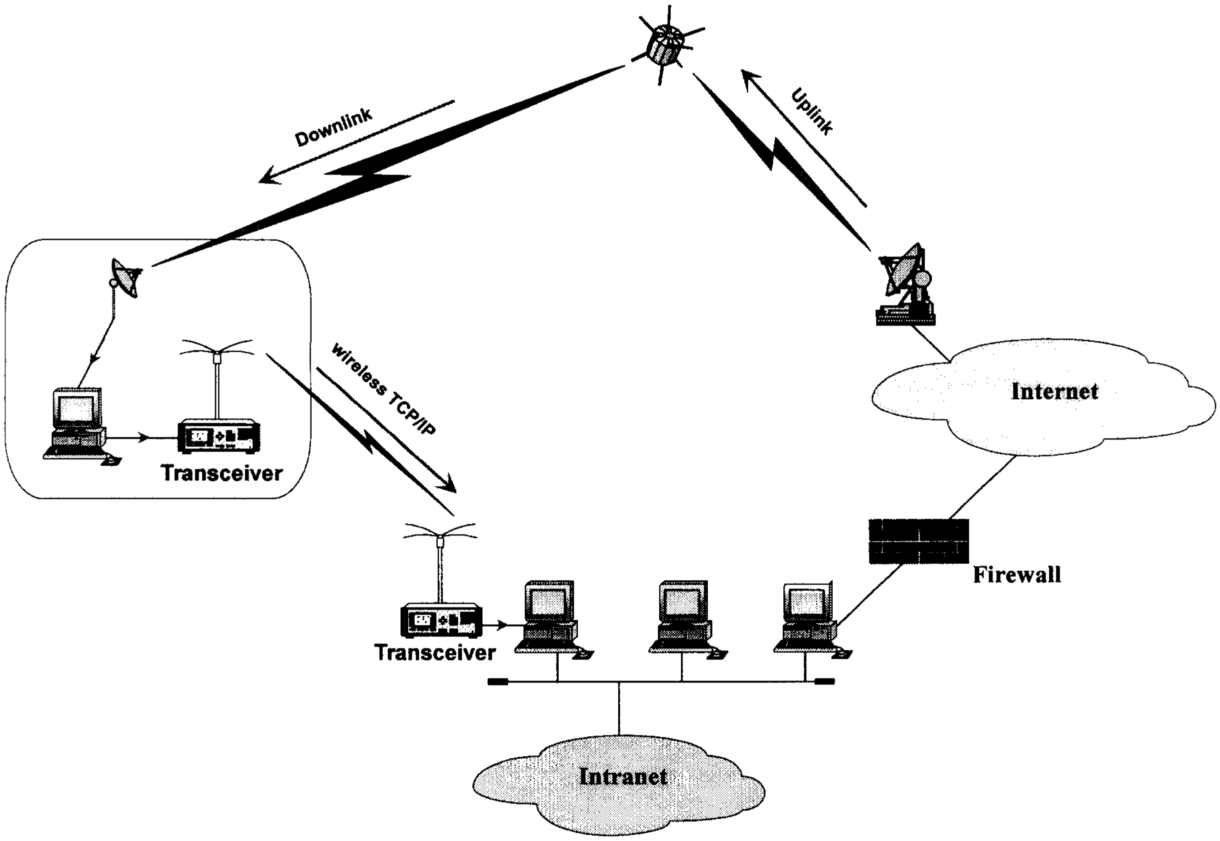


Fig. 5: Combined shortwave satellite transmission with SpaceMan from Rohde & Schwarz

Routing the Internet data stream in this way becomes possible by modifying the Internet protocol (IP), which is responsible for route selection in the Internet. Using what is called IP encapsulation, the IP packages are put into an "envelope" addressed to the operation center. The operation center reads and routes on the envelope contents and, acting as a new user with respect to the addressed Web server, sends the information to the requesting party via satellite. Satellite transmission is unidirectional in this case, ie information can be received but not sent via this path. With Internet requests usually being very short (eg http://www.rsd.de) and the reply data volume comparatively large, the advantages of this technique make themselves felt all the more.

SpaceMan combines the above commercial principle with radiocommunication. Requests to the Internet are made via radio (HF/VHF/UHF) and the help of *PostMan*, and transmission of requested data via fast satellite links. Access to this modern information technology (IT) with radio linkup is realized by means of PostMan, which allows transparent TCP/IP radio data transmission. PostMan in conjunction with shortwave transceivers of the XK2000 family provides unrestricted access to wired communication networks via radio links from any point on the earth. Reception of satellite signals is implemented in SpaceMan by commercial system solutions adapted to radio technology. This provides wireless Internet access unimpeded by the constraints of low data rates.

System components and technology

Apart from the radio equipment, the user requires a dish for the reception of satellite signals and a decoder, which is in the form of an extension card installed in the PC. PostMan together with control software sends user's requests via radio and handles download of data from the Internet to the PC via satellite. In most cases, a commercial elliptical 60 cm dish or similar will do for the reception of satellite signals.

Satellite transmission is via free channels - the socalled transponders - of TV satellites such as ASTRA or EUTELSAT. Data transmission is based on DVB/MPEG2 (digital video broadcasting/MPEG2 is a method for moving picture compression). At the protocol level, a special ADBS (advanced data broadcasting system) extension is used, among other things, to provide filter functions in addition to addressing and routing. ADBS offers various protected access modes (conditional access, security, privacy). This allows individual hardware addressing of any station.

Data rates

The broadband satellite links allow transmission of Internet data at rates up to 400 kbit/s. This is several times the data rate of conventional V.34 modems with max. 56 kbit/s or ISDN with 64 kbit/s. The data rate of 5.4 kbit/s afforded by shortwave appears modest in comparison, but is of little consequence considering that Internet requests are rather short.

Problem-free operation is guaranteed as long as the user is within the footprint of the satellite. This combination of radio and satellite transmission can also be used on ships with the benefit of undreamed-of data rates at low charges.

Safety

The transmitted information is encrypted to protect it against unauthorized interception. In addition, end-to-end encryption provides a high degree of safety.

Conclusion

The approaches described above open up completely new perspectives to mobile users who in the past had no access to wired communications because of poor infrastructure.

COTS based systems: the necessity of a service & systems management strategy to assure service levels

(March 2000)

Dirk Somerling
Tivoli Account Manager
IBM Belgium
Square Victoria Regina 1
B-1210 Brussels, Belgium

COTS based systems create the need for system management strategy: In large military organisations, as in any traditional business organisation, the collection of the tasks to be performed by every employee results in the service the organisation provides. In order to perform his tasks at best, each employee needs a set of tools, which differs according to the task to be performed (e.g. phone, vehicle, etc.). In our particular subject, the employees rely on (a) computer(s) and on the applications and data accessed through or processed by his (several) computer(s).

The kind of applications and data used by each employee will vary according to the role of the employee and the tasks he has to perform, from simple office tools to more advanced workflow or C3 applications, the trend being that the set of applications consists more and more out of COTS. In any case, the employees make use of a mix of critical and non-critical business applications.
In an ideal situation - certainly from an efficiency point of view -, employees should not care about computers, operating systems, underlying protocols or networks. What really matters to them and to the efficiency of the organisation, is the availability of the applications and data in order for them to perform their tasks efficiently at the appropriate time. Inability to perform, whatever the reason, can result in a chain effect - as tasks are interrelated -, creating a negative impact on the performed service of the organisation: its operational capacity can then be impacted to various degrees.

Because of this, it has become very important for IT infrastructures, to create a situation where they can manage the support and the availability of the entire IT capability (from the network infrastructure up to the user interface running on each client desktop), where they can be accountable on the base of service level agreements between the IT department and the users. While in the past (specialised infrastructures), the system management aspects where taken care of by the application from its very first design, the systems based on COTS relies on the assembly of elements not conceived for high availability purposes, hence the need to develop a specific strategy to improve the efficiency and availability of the entire COTS based IT system. Else, the COTS system will be as weak as its weakest link, and the CTRL+ALT+DEL syndrome will apply at critical server levels.

Service & system management through service level agreements: Starting from this perception, Tivoli has over the last years developed a complete set of tools aiming at managing the total IT infrastructure - enterprise system management -, widely accepted by the market. This management platform is the best answer to manage the whole enterprise, reduce the complexity and gain back the control of today's IT environments.

The cornerstone of the efficiency of this platform relies on the fact that it is implemented starting from the business rules of an organisation (e.g. operational requirements) and not from an IT perspective (e.g. driven by the operating system that is being used). Tivoli does focus on managing the IT processes instead of the IT technology. Once the business rules have been defined for a given organisation, they can be applied to the existing IT infrastructure including the COTS (or non-COTS) based applications.

This approach is the only way to increase the efficiency of the organisation and thus provide or increase the appropriate level of availability, reliability and security of the applications and data for each user according to his business needs, hence to the entire organisation. Only then, organisations can provide and achieve high and measurable service levels, be really responsive to business demands and meet the operational goals, even when COTS are being combined as the basic platform for the employees.

You have to gain back the control of the IT environment by implementing systems management best practices. The Problem Management best practice for instance (with the help of pro-active monitoring solutions) will result in shorter problem resolution times and focused interventions with well identified skilled resources. The

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

complete granularity of the Tivoli family of solutions allows to tackle the entire range of systems management disciplines such as problem, change, asset, security, storage, and operations management, and this in a coherent and single interface across the entire COTS stack altogether with the rest of the IT resources (data, applications, hardware, network, users etc.) of the organisation, helping to achieve the appropriate level of service and operationability.

All organisations are confronted with widely distributed heterogeneous systems, interconnected COTS applications and a high rate of change. We should reduce the complexity inherent of today's COTS environments by managing the systems in a consistent way and by simplifying the operations. The management infrastructure does not have to be modified when the migration of COTS (version upgrade or move to another product) occurs because Tivoli is platform independent. The defined business rules will simply be applied and translated to management rules implemented in the newly installed COTS. Organisations are finally able to reach the COTS rollout deadlines and are even less dependent on specific skills to fulfil these operations. This altogether leads to a reduced cost for supporting these regular changes (new implementations or upgrades).

The increased control of the IT resources and the possibility to manage complex environments and applications, is an absolute necessity at the beginning of the e-business era. Obviously, the multiple enterprise systems connected to the Internet need to be managed, with an even higher availability and security, and the operations and processes to support the huge potential growth need to be well managed.

This confirms once more that there is an absolute necessity to implement a dedicated strategy & environment to manage COTS.

Modernizing OMIS, an operational air force C2 system, using COTS hardware and software products

J.G. Stil
ICT Division
National Aerospace Laboratory NLR
P.O. Box 90502
1006 BM Amsterdam
The Netherlands
E-mail: stil@nlr.nl

Abstract

This paper outlines some experiences, gained with the modernization of an existing and operational air force Command and Control (C2) system using Commercial-Off-The-Shelf (COTS) hardware and software products and the adoption of standards, from a practitioners perspective. It describes examples of functional areas where requirements could be met using COTS products alone and where requirements couldn't be met and what strategies were followed to meet these requirements.

1 Introduction

This paper presents an example of the application of COTS hard- and software products for the modernization of an operational air force C2 system. The system consisted of a tailor made application running on COTS hardware. The application has been modernized using as much as possible COTS software products and the hardware has been replaced by leading edge technology hardware. The aim of this modernization was to come to a system based on state of the art technology. The new system had to be easier to use, have capabilities for interoperability, lower maintenance costs and it had to form a solid base for future functional extensions.

The paper describes how COTS products are applied to fulfill the military requirements, with emphasis on the application software. This paper also explains what measures have been taken to satisfy requirements where COTS products alone are not sufficient.

2 COTS products in a military environment

In the last decade more and more COTS information technology (IT) products become employed in military environments. Personal Computers, operating systems, office suites, database management systems, etc., originally meant for the consumer market, appeared to be usable to fulfill the military requirements. Implementation of these leading edge technology products in defense applications has become an attractive alternative for custom made systems and is generally seen as an effective way to cope with reduced budgets and staff. Advances in technology are no longer driven by military applications, but rather the military market only needs to exploit technology that exists in the commercial market [6].

Besides the lower costs, application of commercial available products can result in shortened acquisition times and a shorter *time-to-deployment* and therefore could provide military advantage, as stated in [2] and [7]. The earlier a leading edge technology becomes deployable in the battlefield, the better. The products are available relatively fast and the prices for these products continuously decrease. Furthermore, if products are available from multiple suppliers, dependency on a single supplier reduces.

Note however that in some situations application of COTS products alone is not sufficient, especially when

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

requirements are very military specific. COTS alone often results in a so-called 80 percent solution, which is generally what the COTS solution represents in terms of a comparison toward the customed desired or customed tailored requirements [3].

In many situations the missing requirements can be met by implementing glue code or by making modifications to the COTS product. In other situations it not possible to determine whether requirement can be met with a certain COTS product. In these situations it is often no longer possible to treat the COTS product as a black box. More (inside) details about the product might be necessary [5] and possibly involvement of the manufacturer.

M. Looney and J. Briggs even state: "By definition, any development project cannot be completely COTS. There must always be some glue to integrate components or customize them which implies some level of understanding and involvement" [4].

Following paragraphs describe how COTS are applied in an operational military environment and what measures have been taken to cope with the shortcomings of the applied COTS products.

3 Overview OMIS

The example presented here describes the modernization of OMIS. OMIS is the Operations Management Information System which is in use by the Royal Netherlands Air Force (RNLAF) at Volkel Air Force Base in The Netherlands since 1983. OMIS is a command and control (C2) system which has as main goal to support the RNLAF in its task to prepare aircraft for missions to be flown. OMIS assists in the communication of all necessary information between different control centers and units and provides all users with consistent and up to date information, needed to perform their task. A schematic overview of the OMIS functionality is shown in figure 1.

The system met all requirements with respect to functionality, way of operation and security. The functionality was assured through continuous maintenance and adaptation of the application software to fit the changing needs of Volkel Air Force Base with respect to their business rules. The operational availability was assured by adding redundancy through application of multiple computers which mutually synchronized all information. Security was assured through the implementation of a sophisticated access control mechanism based on *the need to know* principle. The system was approved by military intelligence and was intensively used for the daily operations and during exercises.

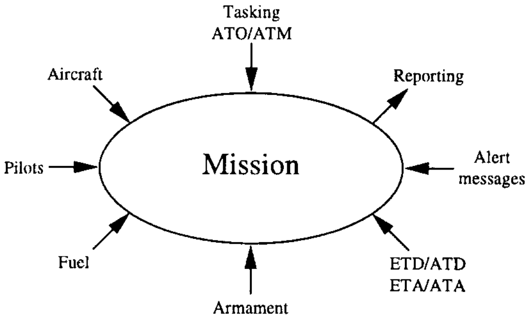


Figure 1: OMIS functionality

OMIS consists of tailor made application software running on COTS hardware which consisted of DEC PDP-11/84 minicomputers, interconnected with each other via DECNET (including crypto devices), and DEC VT-420 terminals (see figure 2). The OMIS application software was developed by the National Aerospace Laboratory (NLR) in the Netherlands, under responsibility of and in close cooperation with the RNLAF.

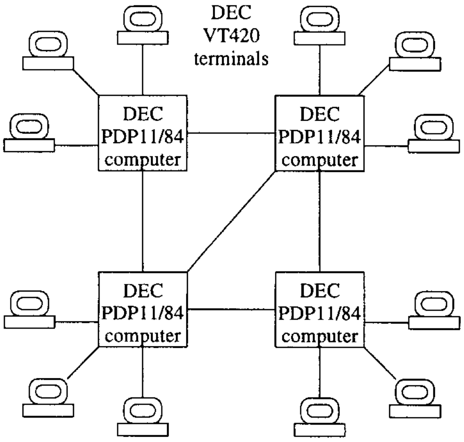


Figure 2: The OMIS network architecture

Not only the application software was tailor made. Functionalities that are less application specific, like database management and data replication, were also tailor made.

4 System life-cycle and modernization

4.1 Life-cycle aspects

In the mid 90's it became clear that it was no longer cost effective to maintain the current OMIS. The main problem was that the hardware had become obsolete. Spare parts became scarce and expensive maintenance contracts where necessary keep the systems up and running. Besides that, the OMIS software was of a former generation of software that lacked standardization and the capabilities for extension and interoperability. However, because the system was constantly adapted to fit the changing needs, the functionality was still valid and absolutely required to support the execution of the daily operations. Officers from Volkel AFB often say: "We can't fly without OMIS".

The RNLAF also had the intention to introduce OMIS at other bases in the Netherlands. Therefore the RNLAF decided to modernize OMIS. The resulting system had to be functional equivalent to the original system and had to meet the same requirements with respect to functionality, way of operation and security.

Further in this document the original OMIS will be called OMIS-1 whereas the modernized version of OMIS will be referred to as OMIS-2.

At the same time the RNLAF decided to realize a complete new IT infrastructure at all their bases. This new infrastructure, called KLuIM, had to be realized using COTS hard- and software products. KLuIM forms an *implementation middle layer* and should be used as the basis for all future applications. This new information infrastructure is intended to provide a multilevel secure environment in which command and control applications and office-like applications are used simultaneously.

Another justification to modernize OMIS was the changing operational role of the RNLAF. Till the late 80's the main task of the RNLAF was the defense of NATO territory during the cold war. This role has now changed to a role in which the RNLAF participates in multinational peace keeping operations, possibly where operational units are temporarily deployed *out-of-area*. This new operational task requires communication with other participating forces and therefore *interoperability* with other forces's information systems.

A technical, but certainly important, argument for modernization was the fact that the OMIS application and the operating system were judged not Y2K compliant.

4.2 Choices made

Two options for the modernization of OMIS were distinguished. The first option was to upgrade the hardware only and run the application software on an up to date platform, using emulators of the original hardware. This option guaranteed the operational continuity (only the Y2K problem had to be solved), but the resulting system still would lack standardization (at least at software level) and capabilities for interoperability and extendibility. Another draw back of this option was that emulators were only supplied by the manufacturer of the original hardware. This would result in a lack of freedom of supplier. This option was considered not very attractive.

The second option was to upgrade to the new COTS products based information infrastructure, existing of a new hardware platform, a new operating system and a new network, and to re-design and implement a functional equivalent of the application software using as much as possible COTS software products and standards. The application would be brought to state-of-the-art technology. The functionality of OMIS itself is defense unique, even air base unique. The type of application however is not defense unique and could be applied to non-defense environments, so it seemed to be possible to apply COTS products.

An argument for the second option is on standards for interoperability. Because the software had to be redesigned, the RNLAF also had the possibility to conform to a standardized data model to enable interoperability. The ATCCIS standard was adopted for OMIS-2 and adapted to the air force situation.

The RNLAF had plans to introduce OMIS-2 at other air bases also and therefore has chosen for the second option because this option offered a better maintainability and more opportunities for future extensions, which were indeed defined.

5 Hardware

5.1 General configuration

The choice of the hardware was, amongst others, influenced by the requirement that it had to be possible to run a wide variety of commercially available office applications and to use OMIS-2 for out-of-area operations. It had to be possible to use a small OMIS-2 configuration at locations where an operational unit of the RNLAF is deployed temporarily, possibly connected with the OMIS-2 configuration at the home base.

A choice was made for an Intel based computer platform running under the Microsoft Windows NT4 operating system. This choice allowed the application of a wide variety of Personal Computers, from laptop to large server, depending on the needed capacity and size of the configuration and a lot of office-like software products are available for an Intel/Windows based computer platform.

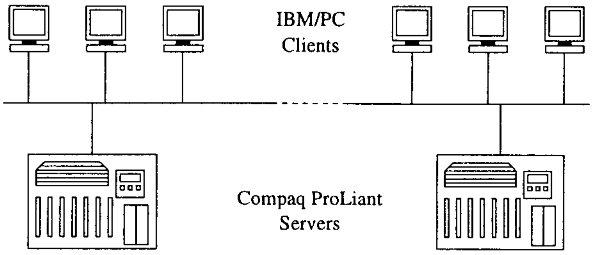


Figure 3: The OMIS-2 network architecture

The client computers are equipped with removable hard disks. This offers the advantage that defective systems can easily be replaced while the workstation related data stays available for the user, under the assumption that no disk failure occurred. Another advantage is that the disks of classified workstations can be locked in a safe outside operating hours.

Choosing de-facto PC standard hardware allows easy system upgrades to meet increasing performance requirements. At the start of the modernization, leading edge hardware could hardly meet the performance requirements. When the system was operationalized, which was about two years later, the advances in hardware technology happened so fast that the performance problems were no issue anymore.

5.2 Security

The new information infrastructure had to provide a multilevel secure environment at the operational air bases in the Netherlands. The need for a multilevel secure environment resulted from the requirement to be able to run office-like and command and control applications in the same environment. However, COTS network encryption hardware was not available due to the absence of military accreditation for these products. Accreditation was absolutely required because OMIS-2 needed to run on NATO SECRET level.

The KLuIM infrastructure was not available on time for the modernized OMIS and therefore the RNLAF decided to build a separate network. This network was realized conforming the standards that were defined for KLuIM and is used for command and control applications only. Later, when the encryption hardware becomes accreditated, this network will be used for office-like applications also.

The network that is implemented is an ATM-switched fiber optic network. For security reasons, not only the backbone is realized using fiber optic equipment but also the end user workstations are connected to the network via fiber optic cables also (no copper cables). This separate network is accreditated to run NATO SECRET.

5.3 Survivability

To assure failsafe operation at hardware level and continuity in emergency situations, the OMIS-2 configuration at Volkel AFB consists of multiple server computers, each located it a secure location. Whenever a server fails or gets lost, the other server(s) take over its tasks, mainly related to operating system and network management, so that the system stays available for the operational users.

Reliability of the servers itself was increased by applying hot pluggable RAID-5 (Redundant Array of Inexpensive Disks) disk units. This technology allows replacement of defective disks without interrupting system operation.

6 Application Software

During the modernization of the software, four important functional areas were encountered where COTS software products alone were not sufficient to fulfill the OMIS-2 requirements. These areas were security (especially access control), logging, survivability and interoperability. Following paragraphs describe what measures have been taken to satisfy the requirements.

6.1 General approach

General approach was to apply as much as possible COTS software products to meet all requirements. Some requirements that couldn't be met by the application of COTS products alone were satisfied by implementing missing capabilities on top of the COTS products using COTS available development tools. Other requirements could be satisfied by tailoring parts of the COTS software products.

The new OMIS had to be a client-server application in which all user interface related functionalities were performed by clients and data management activities mainly by the server(s). The data stored in the servers had to be structured in a model compliant with the ATCCIS standard model.

Data management requirements could be satisfied by commercial available Relational Database Management System (RDBMS) products from Oracle. Database design tools from Oracle provide design capabilities compliant with ATCCIS modeling techniques.

In the future it might be possible that the applied COTS software products have to be upgraded to newer versions. To minimize the risk that the newer versions are no longer compatible with tailor made parts of the system, only capabilities of the COTS software products are used that are not de-supported by the manufacturer. Hardware and operating system specific features were avoided completely.

The original OMIS software contained some capabilities, especially related to security, logging and survivability which could not be provided by COTS software products alone. These issues will be detailed next.

6.2 Security

OMIS-2 required an access control mechanism following the *need to know* principle. This mechanism had to control the access to particular parts of the application and to the data accessed by those parts. Classification levels of the user and the location of the workstation had to be taken into account also when determining whether access was allowed or not. For example, parts of the application might only be accessible from a workstation located in a secure place such as a bunker.

The access control data had to be available on all participating database servers and be consistent. In a typical OMIS-2 configuration, multiple server computers are applied to assure continuity in case a server becomes unavailable. This means that users must have access to more than one server using the same user name/password combination.

Modifications to the access control data had to be made via a *two-men concept* to prevent security violations by administrators.

The access control mechanism provided by Oracle7 is only based on rights to access data stored in the database. Access to specific parts of a client application couldn't be controlled by this mechanism directly. This problem was solved by implementing a custom access control mechanism on top of the standard Oracle7 mechanism. This mechanism is used by the client application to determine access to the different parts of the application.

Standard Oracle7 also did not provide facilities to maintain a centralized user administration for distributed and replicated database servers. This problem was solved by implementing a mechanism which periodically synchronizes the distributed user administrations of the multiple server computers.

The problem related to the maintenance of the authorization related data was solved by implementing a special authorization data maintenance application using the COTS software development tools. This application forces administrators to apply changes on the authorization related data via the *two-men concept*. This means that modifications have to be made twice, by two different operators and within a certain time frame. After the first administrator has made modifications to the authorization data, this little application temporarily stores the modified authorization data in the database, so it can be used for comparison when the second administrator makes the same modifications. Only when the modifications, made by both administrators within the preset time frame, are exactly the same, the modification is accepted.

6.3 Logging

The original OMIS used a very extensive logging mechanism. For all modifications made to data in the database, the old and new values, the user making the modification and the time the modification was made, were registered. Besides a log of data mutations, an event log was maintained to register user actions. Both logs offered the capability to reconstruct the series of activities and mutations in case of system malfunctioning or security violations.

OMIS-2 required a similar logging mechanism to register all data manipulations and user activity. The mechanism provided by the Oracle7 database server (*tracing*) was not suitable for OMIS-2 because at some levels it did provide too detailed information whilst at other levels it did not.

This problem was solved by implementing a simple logging mechanism in the databases. This logging mechanism is based on a generator which generates logging facilities for specific data sets. By applying this technique the logging subsystem can easily be updated whenever changes to the structure of a data set have to be made.

6.4 Survivability

In OMIS, survivability was assured by applying multiple computers, each with a complete set of data stored on it. A tailor made replication mechanism kept the data on the different computers synchronized. For OMIS-2 this functionality had to be realized using Oracle capabilities.

Standard Oracle7 provides mechanisms to setup distributed databases and for database replication. The database replication mechanism takes care of the distribution of modifications made on one server to the other servers participating in a replicated environment. The replication mechanism provides facilities to detect conflicting simultaneous data manipulations on separate servers and methods to solve these conflicts.

The standard mechanisms couldn't be applied directly because the conflict detection techniques did not allow simultaneous updates on different attributes of the same object which was absolutely required by OMIS. For example, it had to be possible for a logistic officer and an operations officer to assign an aircraft and a pilot respectively to the same mission simultaneously when connected to different server computers.

The Oracle7 replication mechanism was slightly modified so that above mentioned operations could be performed. Also the conflict detection and resolution mechanism needed some simple modifications. The modifications made to the standard software are temporary since newer versions of Oracle ($\geq$Oracle8) provide required capabilities standard.

6.5 Interoperability

A new requirement for OMIS-2 was the capability to interoperate with other C2 systems. For OMIS-1 there was no such requirement and therefore this system lacked capabilities to interoperate.

At network level the interoperability requirement was met via the application of standard network hardware and software. At application level this requirement resulted in a complete re-design of the data model. The ATCCIS standard data model was used as basis for the new data model. All entities in the OMIS-2 functional environment were re-analyzed, normalized and placed in a so-called *ATCCIS-able* data model. Adoption of the ATCCIS concept facilitates future coupling with other national and possibly international (COTS based) Command and Control systems that are based on the ATCCIS model.

Application of a COTS relational database management system offered the possibility to utilize leading edge database technology for OMIS-2. The Oracle Relational Database Management System provided the enough functionality to implement the new data model. Database design tools from Oracle were used for the design of the database. These tools allowed automatic generation of the database.

7 Concluding Remarks

The modernization of OMIS showed the successful application of COTS products for a functional re-hosting. The re-hosting resulted in a system with a 100 percent equal functionality, but based on leading edge technology and with improved capabilities for future extensions and an improved ease of operation and management.

The application of standard PC hardware for the modernized OMIS showed that an assurance level at least equal to the assurance level of the original OMIS is possible.

The presented example showed that not all functionality could be realized directly by the COTS products itself. It appeared not to be possible to meet requirements related to security, logging and survivability using COTS products alone. These requirements were satisfied by implementing small modifications to the COTS products or by successfully using applying COTS software development tools to implement missing functionalities.

The requirement for interoperability was satisfied by using the ATCCIS standard for the data model. The resulting data model was implemented using COTS data management products without any problem.

Mid 1999, OMIS-2 is installed and operationalized at Volkel Air Force Base in the Netherlands. The configuration consists of multiple servers placed at secure locations, and client workstations all over the base. Minor problems were encountered during the installation, mostly related to the scaling of the system. After the installation it took only three days to operationalize OMIS-2. From mid October 1999 the modernized OMIS runs smoothly. Intensive usage during large exercises did not result in problems.

References

[1] *First Workshop on Ensuring Successful COTS Development*, May 22 1999.

[2] Emmett Paige Jr. (assistant defense secretary for command, control, communications and intelli-

gence). Striving for Information Superiority. *Defense Issue*, 11(72), June 22 1996. Prepared remarks to the 311th Theatre Signal Command Activation Dinner.

[3] Lloyd Mosemann (deputy assistant secretary of the air force). COTS in the Air Force - Success Story. 1995. Prepared statement to COTS'95 Conference.

[4] Micheal Looney and Jim Briggs. Some experiences and comments on the use of COTS Software in UK Naval Combat Systems. In *First Workshop on Ensuring Successful COTS Development* [1].

[5] Norman F. Schneidewind and Allen P. Nikora. Issues and Methods for Assessing COTS Reliability, Maintainability and Availability. In *First Workshop on Ensuring Successful COTS Development* [1].

[6] Richard Scott. Putting COTS in command. *Jane's Navy International*, 102(4), May 01 1997.

[7] Paul G. Kaminski (undersecretary of defense for acquisition and technology). Investing in Tomorrow's Technology Today. *Defense Issue*, 10(46), March 28 1995. Prepared statement to the Research and Development Subcommittee, House National Committee.

Experiences in designing radio monitoring systems using Commercial Off-The-Shelf (COTS) components

Dipl.-Ing. Günter Palten
Rohde & Schwarz GmbH & Co KG
Radiomonitoring and Radiolocation Division
P.O. Box 80 14 69
Mühldorfstraße 15
D-81614 München

Abstract

As military tasks become more and more complex budget will increasingly be limited due to the national economic demands. Simultaneously customer specific requirements on near real-time processing, high availability, tailored systems and integrability into NATO Interoperability Management Policy are growing. The new challenge for developers and designers on the one hand consists in meeting these customer requirements and in offering modular and flexible components which can be integrated into legacy systems. On the other hand development costs have to be reduced and the time for assembling and delivering systems have to be shortened. Therefore systems for military purposes have to integrate and have to be developed with more and more extendable and pre-built standard Commercial Off-The-Shelf (COTS) components.
As a main partner of the German armed forces the Rohde & Schwarz Radiomonitoring and Radiolocation Division offers customer tailored and component-based systems as well as system integration services using software and hardware COTS components. A lot of experience has been made during that process of system development, tailoring and integration using COTS products.

The Rohde & Schwarz radiomonitoring systems may be seen as one of these numerous examples for integrating COTS together with customer specific components. These systems also show the effects the use of COTS products may have on procurement and development process and system architecture. Radiomonitoring systems by Rohde & Schwarz represent a concept which meet the requirements of a state-of-the-art monitoring, location and analysis system. They are built up in a highly modular way and developed, built with and grouped around typical COTS products like commercial data bases, interfaces and hardware and software modules. The software architecture and the system concept provide client-server functionality and links to complementary products such as frequency management software, geographical information systems (GIS) and analysis systems. Custom-tailored solutions are manufactured by connecting standard hardware components and off-the-shelf, tested software modules.
Due to the modular concept radiomonitoring systems can easily be upgraded from a compact to a more complex and interoperable system. Standard interfaces allow high communication connectivity within local or world-wide networks and support therefore required interoperability with coalition and legacy systems.
The present paper describes experiences using COTS components and forming functional systems from software and hardware integrants by adapting them to customer specific requirements.

1. Introduction

Due to the progress in defense technology modern systems for military purpose are becoming more and more complex and increasingly expensive. Numerous components have to be integrated and have to align with requirements like integrability into legacy systems, interoperability and system capability. In times budget is usually limited, customers and industry have two different aims. Users and customers expect cost-effective systems which cover every requirement and which can be integrated into an existing system environment without any additional costs. System designers and developers have to be as individual as possible to give the customer the impression of uniqueness. Simultaneously they have not to

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

invent really new systems with every commission they are working on. System designers and developers have to keep a large number of industrial and specific standards, protocols and techniques, use state-of-the art methods, tools and devices each one of which needs to become familiar with.

They therefore integrate more and more off-the-shelf products, hardware components and devices and use often COTS hardware and software components to keep close to the customers needs or to reduce the effort in the own software development process.

But with the use of third party products – commonly defined as COTS components – and the integration into a system miscellaneous problems raise than building a system with completely own products which have been developed and constructed presently or previously internally by the development organization itself.

Being aware of these problems we have decided to use COTS products for building customer specific and tailored systems for radio surveillance and monitoring purposes as well for military as civil clients. Rohde & Schwarz is looking at these problems from the integrators point of view, who is using COTS products to build systems rather than from the COTS builders point of view himself. The following paper is a report about the experiences we have made, concerning integration of COTS products in systems for military and intelligence purpose.

Using radiomonitoring systems as an example the paper describes the approach and the possible points to bring in these products. Starting with a basic overview on radiomonitoring systems, their components, functionality and operational structures points of attachements will be shown for the use of COTS. Experiences out of the development and integration process will be outlined. For the process itself had not yet been finished a final statement about the success can not been made.

2. Definiton

From the system developers‘ and integrators‘ point of view Commercial Off-The-Shelf (COTS) products are commonly defined [1,2,..] as components provided by a third-party vendor. On the one hand they may be used for the development of a system, on the other hand as a hardware or software component of the system itself. By definition they are products which have to be accepted as they are, because system builders have little or no influence on maintenance and evaluation. Commonly they behave and must be treated like a black box. Another characteristic and important feature among a lot of others is the grade of evaluation and the wide spectrum of customers and tasks they are designed for.

But exactly these characteristics cause the different set of problems but also cause challenges typical system integrators have to deal with.

3. Background

A first step in designing and building a system for radio surveillance and monitoring by integrating COTS products is to understand these systems and its functional concurrence in its entirety. The system technique itself and the reproduction of procedural steps into a modern workflow is nowadays an important component of state-of-the art system integration. But not only the technical design of these system change rapidly also the required and realised concepts are modified. Reasons are the increasing development and improvements in defence and communication technology on the one hand and in user requirements like efficency and user - friendliness growing with the pretentious tasks on the other hand.

Nowadays realization of these requirements is nearly unthinkable without using modern computers, user-specific software and software tools. For several reasons – cost efficiency, modularity, reuseability, standardization - also COTS products play an important role within radio monitoring systems.

Modern radio monitoring systems as they are developed and manufactured by Rohde & Schwarz consist of numerous components and subsystems functionally working together. Components of each systems are computer hardware and software as well as mainly computer-controlled special equipment such as antennas, receivers, analyzers and direction finders. Mostly they are systems with automated features and monitoring, analysing and visualization capability. They contain distributed, intelligent subsystems for measurement and location of electromagnetic emissions working close together within the system. Also the workflow and tasks on the entire monitoring process, the flow of information and the information management

within the system play an important role when mapping them into a suitable software.
Measurement results may be stored in data bases and processed by powerfull computers with analyzing tools and reporting features.
In the term of radio monitoring all processes of acquisition, monitoring and surveillance are included. This comprises also automatic an unmanned surveillance of known emitters as well as identification and surveillance of unknown emitters with an increasing means of analyzing, post-processing and finally reporting on the present situation up to the presentation of tactical situations for the decision process of military leaders.

3.1. Tasks and possible structure of a radiomonitoring system

The task of a modern radiomonitoring system within the complex of electromagnetic emission is shown simplified in figure 1. Herein tasks and functions are hierarchically structured from the process of signal collection (acquisition, search, monitor, bear and locate emitters), over pre-analyzing up to the process of command and control including reporting and processing.

Within the whole signal scenario and electromagnetic spectrum only a small part of the emission has to be acquired and monitored in nearly real time. But exactly this part has to be monitored and processed very fast. Such a task may only be solved successfully by support of computers and modern information technology.

As in many other systems also in radiomonitoring systems a lot of functions and subsystems are working close together and may only be controlled by powerfull computer hard– and software. Thereby the operating staff shall be relieved in routine tasks but also supported in decisions and preparation of suitable means.

3.2. Operational aspects

To find a possible approach of COTS products before system design and development an exact analysis of the users requirement including the used workflow and operational procedures has to be done.

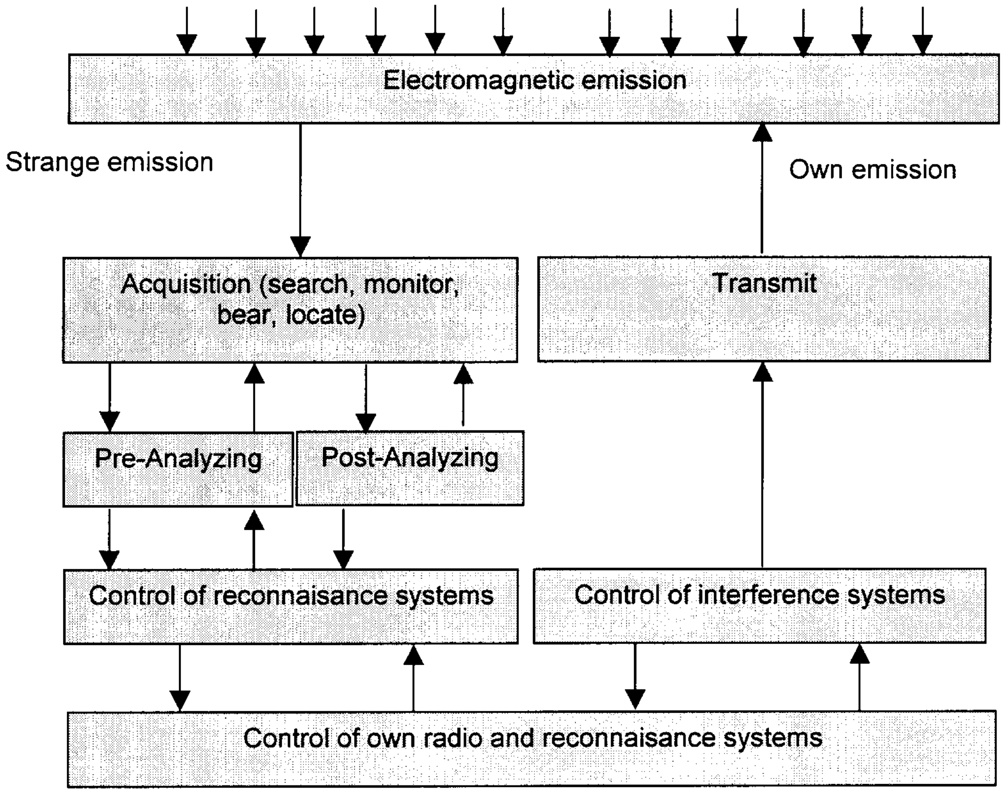

Electromagnetic emission
Strange emission
Own emission
Acquisition (search, monitor, bear, locate)
Transmit
Pre-Analyzing
Post-Analyzing
Control of reconnaisance systems
Control of interference systems
Control of own radio and reconnaisance systems

Figure 1: Task structure within the complex of electomagnetic emission

Derived from the basic functions of a radio monitoring system like

- Receiving operational orders and translating into tasks
- Spectrum surveillance and supervising of known emitters and frequencies. Reporting upon their activity.
- Searching, identification and analyzation of unknown emitters and signals
- recording of signals and analysis
- emitter location
- tactical evaluation and reporting

many tasks have been automated by powerfull computers and special software tools.

The following considerations are based on the possible structure of a modern radiomonitoring system as shown in figure 2. This system mainly consists of different sensoric components (antennas, receivers), signal distribution units, analyzers, control and data processing units and system and database servers. For direction finding and bearing purposes remote DF-sites may be attached via a modern communication link.

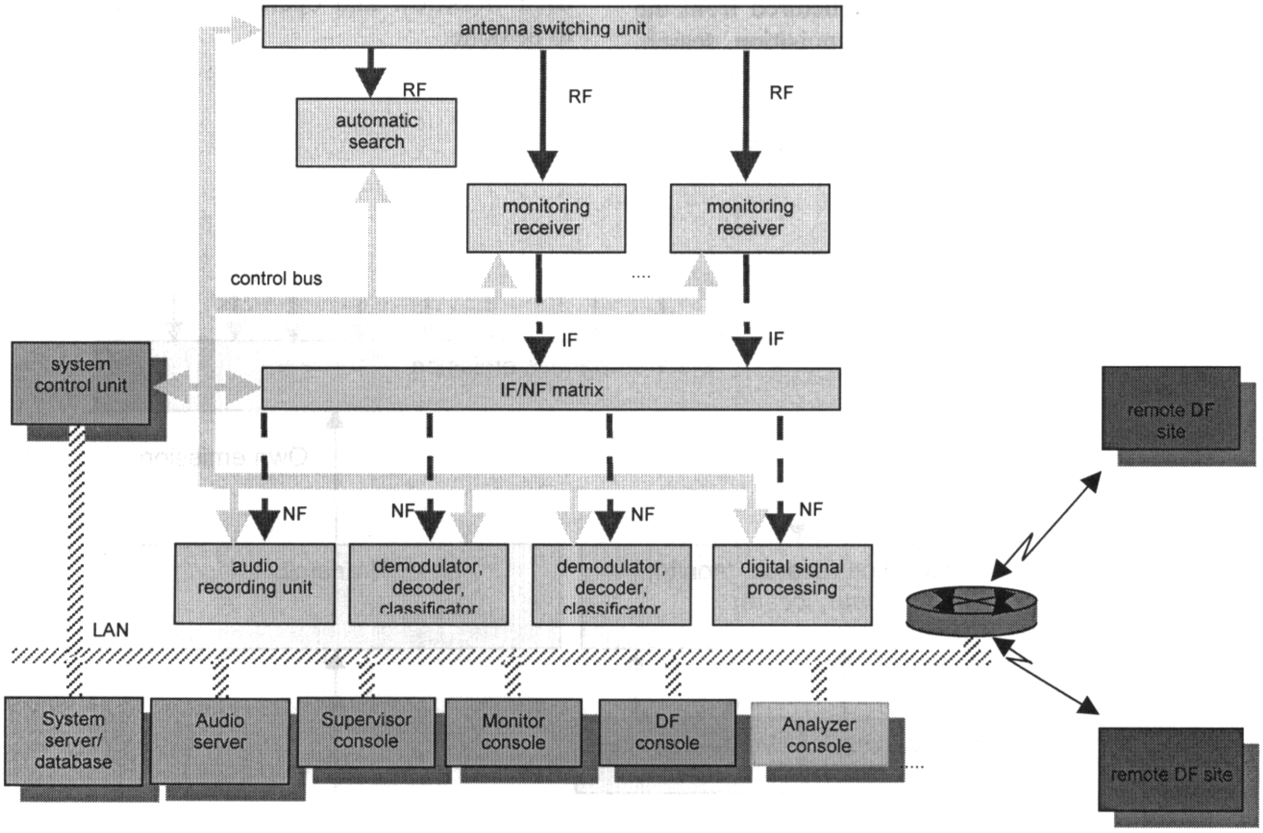


Figure 2: Possible structure of a modern radio monitoring system

The signals of interest will be received via the according antenna and distributed to the dedicated receivers. Then a split up of the signal onto the different hardware tools, like demodulators, decoders and digital signal processors follows. (The tasks of these subsystems is not part of the presentation) On any operator console different tasks, control (analysis and reporting) functions are realized. The allocation of tasks and positions/consoles usually may as follows:

Supervisor: Control and tasking of monitor/search positions and analyzer positions. Report to higher authorities as well as receiving orders from higher authorities.

Monitoring console:
Signal monitoring and recording in a certain frequency band, creating DF requests and reporting on signals of interest

DF console: Search and location of emitters

Analyzer console:
Analyzing unidentified and unknown signals, measurement of signals.

Parts of the system and several operator consoles may be grouped together or extended according to the specific function or the requirements.

The software used within the system has to be of the following kind according to the tasks mentioned above:

- Control software/drivers for specific hardware
- Analyzing and evaluation software
- Reporting software
- GIS and map editing software
- Database
- Communication software
- Translation software.

Recent developments in computer integration and technology, measurement and control equipment as well as in COTS products nowadays allows an extensive grade of automatiion. Thereby processing speed and efficiency may be increased in a perceptible way to support the operator‘s work. Simultaneously costs for system development and integration may be cut.

3.3. Functions to be realized

First approaches may be derived from the functional diagram of a radiomonitoring system.
The usability of common hardware products as antennas, receivers, computers, network and infrastructure components are obvious. They have to be clearly defined and tested for their integrability.
Furthermore software products have then to cover the following essential tasks:

- control the search process (i.e. a channel wise search for frequencies controled via frequeny lists etc. generated by a database referenced scan orders) by a control of the connected devices
- control of the identification process (i.e. forwarding an active channel to the monitoring receivers triggered by certain events)
- system ressource management
- generating automatic requests for direction finding according to pre-defined events
- support of signal analysis, demodulation and decoding (digital signal processing)
- automatic reporting on active channels/frequencies nets etc.
- taking over of DF results and visualization on maps as line of bearing
- generating map based status reports.

4. Example for the integration of COTS products

Since Rohde & Schwarz offers turn-key solutions for radiomonitoring systems, we have experienced with several possibilities to bring in COTS products. Their effective use requires a certain effort to identify where in the system which COTS products can be used in a reasonable relationship to the expense the system integrators have. Having analyzed the functionality of the system, the kind of applicable products have to be defined following a catalogue of criterias for component selection.
Figure 3 basically gives an overview on the system components of and the tasks to be done in

a radio monitoring system in connection with the information and processes to be handled. We have used this diagram to make an estimation where COTS components may be integrated depending on the grade of specification.

Looking on the process of signal acquisition, signal processing and analyzing several in-house and external standard products, like antenna systems, receiver, DSP's and spectrum-analyzer fulfill the COTS criteria. They are not specifically customer-designed but become unique from the time they will be integrated in a system and have to serve for specific tasks. From the moment of adapting these products to specific workflows or customer processes they have to be controlled by suitable software.

Mostly results of software controlled measurements will be the input information for the following hardware and have to be handed over to another device via a standard interface and a standard format.

Specific algorithms are used for demodulation, decoding or signal analysis. Parts of the measurement have to be visualized and serve for operators decision. In this part of the system the share COTS could have is, from our experience relatively small. To guarantee an optimum of integration numerous device drivers and measurement and control software have to be adapted to the customers requirements. Many specific applications resulting out of the requirements and interfaces to almost existing software or databases keeping the customers datasets is a challenge for our system integrators.

The more specific and unique the requirements are the harder it is to use standard applications and COTS products. Therefore we decided to use COTS products only as tools to develop our own measurement and control software and our own device drivers. Besides we add COTS products like GIS software and relational databases to deliver a system covering all the required functions.

When at the end of the process chain the hardware and the functions within a radio monitoring system become more common and the preprocessing provides more standard format outputs an integration of COTS products is simplified.

To generate tactical evaluations and status reports out of the condensed outputs of the sensoric and signal processing hardware a set of standard applications exist which can be used for these purposes. One example is reporting software which is adaptable to standard file formats and offer a standard interface to many third-party COTS products like Office software, GIS applications and relational or desktop databases. To forward these reports we try to integrate even standard applications (eMail etc.) running on standard COTS hardware. From the hardware perspective also for the interconnection of the several operator consoles and sites via LAN or WAN regular network and communication components can be integrated.

To sum up it can be said that in radio monitoring systems COTS products are easy to integrate if the task of that product keeps as common to standards and as simple as possible. As soon as the task or part of the system becomes specific or complex integration and use of COTS products becomes difficult due to the characteristics of COTS. Adapting COTS soft- and hardware to the desired functionality and interfaces of rather complex systems is often more difficult and ineffectice than to generate own software in small edition.

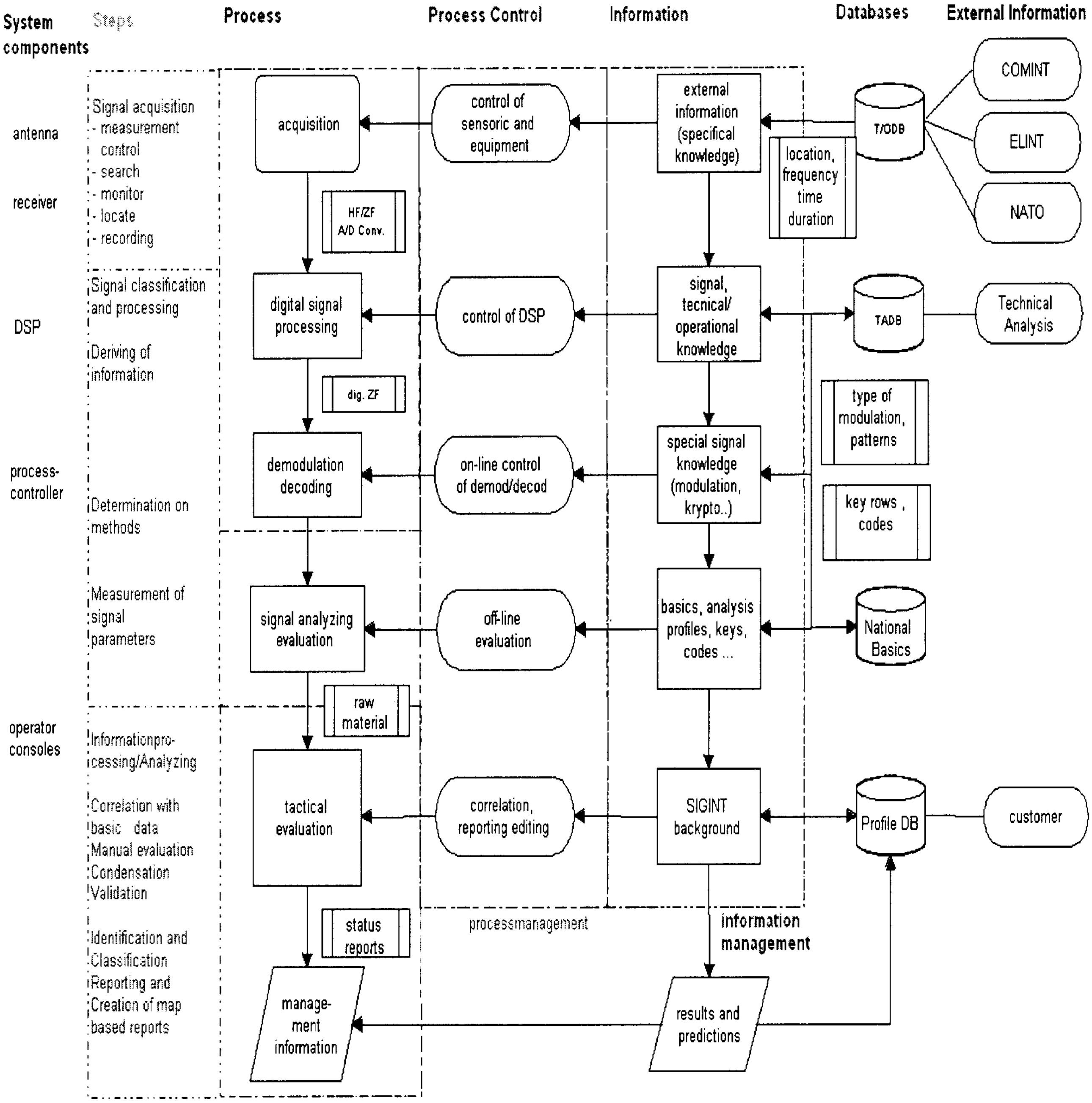


Figure 3: Possible cooperation of system components, processes and information as a basic for COTS integration possibilities

5. Experiences in development and implementation using COTS products

Taking into account the basic structure of a radio monitoring system as described in chapter 3, its operational aspects and the possibilities to bring in COTS we have identified two different perspectives to face that challenge. The system integrator‘s view is more based on the decision how and where in the system COTS products may be integrated in the most cost effective way whereas the developers view reflects the possible use of COTS tools and the question how to generate a COTS product itself to keep the effort for development and reuse as small as possible. Common to both views is that COTS products have certain properties which affect the system and its functionality and therefore have to be considered in time.
After a first process of determining the applicable COTS products the most available ones have to be chosen following certain selecting criteria.

5.1. Example of using COTS for software development

When we have recognized the basic useability of COTS for a customer specific radio monitoring system we looked for a suitable project to start with. Within the complexity of measurement control, monitoring, reporting and evaluation software we picked out the integration of two software products. As a result of the process of integration a new, modular built up software should be created with a unique kernel and different, preferebly COTS based modules.

The first software is mainly determined for the civil client. It is a radio and spectrum monitoring software package which is used to maintain the quality of the spectrum by detecting interferences from licensed and non-licensed users (national and international) and man-made interferences. This software is part of the radio monitoring systems Rohde & Schwarz is building for different customers like public authorities and frequency allocation boards.
The relevant rules and recommendations for spectrum monitoring and spectrum management and the related software behind are from the International Telecommunication Union (ITU), Geneva.

Due to the above mentioned customers as

- broadcast and TV organisations
- ATC (Air Traffic Control) organisations
- frequency regulation authorities

the tasks and features vary from

- long term monitoring of transmitters
- checking of optimal coverage
- providing interference free operations
- getting direction / location of aircrafts / unwanted emissions
- checking of frequency spectrum

up to

- planning of communication links and frequencies.

Especially for the non-civil customers like

- defence forces
- security organisations

and

- law enforcement agencies

another radio monitoring software was originally designed for.

These customers set the focus on tasks like

- searching for known/unknown signals
- monitoring frequencies
- identifying signal sources
- DF and locating signal sources
- getting information about communication nets as
  - station identifiers (call signs / names / numbers)
  - transmission start/stop time
  - directions / locations
  - signal contents
  - technical signal parameters...
- evaluation of monitoring results and reporting.

The aim behind the development was to build a product which offers the common features of both applications and provides enough modularity to cover the civil and military market with additional specific extensions.

By creating own off-the-shelf modules, reducing production and delivery time, overlaying an expandable strategy and designing for stationary, (semi)mobile and network use, the product should have enough flexibility to serve also a wide band client.

5.2. Defining the requirements

A first step to transform the idea into a practicable solution was to define the basic requirements and to specify the key features as

- use of a flexible and modular concept which allows easy adaption to new operational procedures
- separation of functionality and desktop
- definition of external interfaces
- easy installation and maintain
- keeping low development costs and time.

The functional and operational requirements were gathered in a very high level of abstraction and should help us to define the context in which the software will be used and the major functions that it will provide.

The fundamential architecture behind should keep close to a layer based model and contain clearly defined interfaces to relational or desktop databases, networks, devices and users (MMI). Besides it should reflect the major requirements within functionally grouped subsystems and modules.

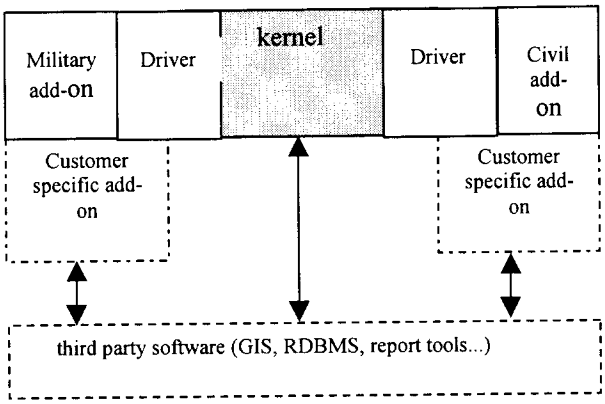
5.3. Setting up the phases

Once selected the suitable COTS tools we started to divide the process into several phases according to common rules for software development:

- Analysis phase (by using use cases to capture the customers requirements and transform them into software functions)
- Design phase (using class diagrams)
- Implementation phase (code generation with C++, verification and validation)
- Test phase (module, subsystem and interface testing using specific test tools).

5.4. The experiences

The first phase of prototype implementation consists of building a basic subsystem as a functional kernel covering the different customers' requirements. Specific add-ons should allow us to customize the software due to the specific demands.



Within that subsystem we started with a standard device driver module for receiver control with interfaces to hardware and GUI components and a LAN based audio recording/playback module with interfaces to a COTS database.
With the use of standard products like Use Cases, class libraries, standard software development tools and according to object oriented software development standards we tried to build a first prototype. It should provide a basic functionality and contain almost sufficient functionality to interact with the according hardware devices within the radio monitoring system.

Already during the phases of development experiences described in other publications [1,2] could be confirmed. A lot of properties of COTS supported software development and integration of COTS components became obvious. Most of the experiences we dealt with concern the interface between own modules and source code and COTS components.

Because of the fast evolution of COTS products a clearly defined interface is absolutely necessary. The architecture of the generated software must allow an isolation of COTS products. Otherwise an expansion of the final product or a complete substitution of COTS parts is nearly impossible. One of the main problem we had to deal with was the near real time processing of audio data within the network. Due to the different requirements of the customer to the system performance we had to try several versions of drivers to reduce the delay time to a minimum.

Furthermore clearly defined and possibly standard interfaces makes developers and finally customers independent from proprietery COTS products. As an example we used class libraries from Roughware for Windows NT ™ and Unix systems to be open for different operation platforms and to reduce the development process for both systems.

SQL and ODBC interfaces allow access and data transfer from/to external databases such as Oracle or MS Access. The use of customer specific data may be guaranteed.

Of essential importance was the test of the used and generated modules with test tools.

Derived from these experiences we figured out the major steps to follow when using COTS components in our radio monitoring systems.

The qualification process starts with identifying the properties of a component and its qualification for the intended requirement. This includes items as functionality, use of standard interfaces, reliability and exchangeability in case of adaption to changed requirements etc.. Especially for the process of development the testing and reusing is another major factor for the use of tools and components as part of a whole.

But nevertheless also external dependencies played an important role when we decided for certain COTS components. Because of the frequently update process of COTS, integrators have to face additional risks when using these products. Due to the life cycle of our products a simple update of a single component like an upgrade of a new data base version is realistic. But this may have other effects within the whole system and result in malfunctions especially in time- and data-critical applications.

Through the process of assembling COTS into our own software we have seen that the interface and data exchange structure is almost important. This may influence the portability and interoperability of the system. Here some styles cristallized like:

- The centralized style which is based on a common database and shares information via this information pool.
- The message handling style in which each component has its own data store and data transfer is coordinated by messages or procedure calls of the components.
- The object oriented style in which Broker provides mechanism for object location and activation.

Considering the characteristic customer who already has long grown centralized data like frequency or operational data bases we concentrated on the first style.

From the aspect of exchangeability we had to be careful with the simplistic view of upgrading and replacing components during the phase of development and integration. Replacement of components often was a very difficult and time-consuming process due to the mostly non-identical successors, different behaviour and resulting test phase.

As a result of our COTS experiences we have recognized that a structured planning and definition of the use of COTS and its purpose in the system makes the whole process of development and integration easier. Besides the questions for qualification, reuse, functionality and interoperability also long term considerations and usage play an important role.

The costs often considered as one main factor for using low price COTS instead of own products is on the second view not as significant as it was originally. During the implememtation of the first systems additional costs for customer training, maintenance, licensing and error tracking and correction occured. This reduces the price advantage of COTS products often to a minimum.

6. Conclusion

A lot of properties of COTS supported software development and integration of COTS components became obvious during our software project. As a conclusion the experiences are summed up thematically grouped from integration via coding to maintenance and testing.

Integrability:

- When integrating COTS products a reasonable relation between effort of integration and adaption of own products and the requirements one wants to cover with COTS must exist (unfortunately this is not foreseeable when starting the development and deciding for a product).
- Furthermore clearly defined and possibly standard interfaces makes developers and finally customers independent from proprietery COTS products. As an example we used class libraries from Roughware for Windows NT ™ and Unix systems to be open for different operation platforms and to reduce the development process for both systems. SQL and ODBC interfaces allow access and data transfer from/to external databases such as Oracle or MS Access. The use of customer specific data may be guaranteed.
- Because of the fast evolution of COTS products seperability is absolutely necessary at each time. The architecture of the generated software must allow an isolation of COTS products. Otherwise an expansion of the final product or a complete substitution of COTS parts is nearly impossible.

Coding:

- Because of the evolution process of COTS the market requires, the own coding effort has to be aware of this evolution and to watch for new version and releases which could suddenly become interoperable with the code the devloper has written yet.
- COTS components affect the functionality of the whole software for instance in time-critical applications (essentially when time errors occur while one device or module is waiting for data of another application).
- Once the system integrator has decided to use a certain COTS product, a necessary upgrade of own products has to wait unless also the COTS product will be upgraded too. This reduces the evaluation process of the own product.

But we also made the positive experience that several tools support the development process and COTS may extend the functionality of the own product by extending its possibility (like evaluation, map processing and GIS software)

Maintenance and testing:

- Because of the fast evolution of COTS software a maintenace is rather difficult and the system integrator has to keep enough qualified personal on hand to be able to integrate and use the current version.
- Furthermore a configuration management is absolutely necessary to cover all releases
- Integrators and developers should use appropriate test tools during the development to guarantee the interoperability to their piece of software at any time.

References

[1] Dr. M.R. Vidger and J. Dean
An Architectural Approach to Building Systems from COTS Software Components.
NRC Report Number 40221

[2] Dr. M.R. Vidger and J. Dean
System Implementation Using Commercial Off-The-Shelf (COTS) Software.
NRC Report Number 40173

[3] R. Grabau
Technische Aufklärung.
Franckh'sche Verölagshandlung, Stuttgart

[4] Dr. U. Ullrich
Systemtechnik für die Funküberwachung und Funkaufklärung.
In Neues von Rohde & Schwarz special, Seite 32-38.

REPORT DOCUMENTATION PAGE

1. Recipient's Reference	2. Originator's References	3. Further Reference	4. Security Classification of Document
	RTO-MP-048 AC/323(IST)TP/7	ISBN 92-837-1049-5	UNCLASSIFIED/ UNLIMITED

5. Originator Research and Technology Organization
North Atlantic Treaty Organization
BP 25, 7 rue Ancelle, F-92201 Neuilly-sur-Seine Cedex, France

6. Title

Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"

7. Presented at/sponsored by

the Information Systems Technology Panel (IST) Symposium held in Brussels, Belgium, 3-5 April 2000.

8. Author(s)/Editor(s)	9. Date
Multiple	December 2000

10. Author's/Editor's Address	11. Pages
Multiple	210

12. Distribution Statement There are no restrictions on the distribution of this document. Information about the availability of this and other RTO unclassified publications is given on the back cover.

13. Keywords/Descriptors

COTS (Commercial Off-The-Shelf)
Computer systems programs
Computer systems hardware
Software management
Software maintenance
Life cycle costs
Military applications
Quality assurance
Evaluation
Safety
Reliability
Procurement
Design
Tests
Verifying
Proving
Adaptation
Standards
Interoperability
Integrated systems
Upgrading
Obsolescence
Sustainability

14. Abstract

This volume contains 24 unlimited papers and 2 Keynote Addresses presented at the Information Systems Technology Panel Symposium held in Brussels, Belgium, 3-5 April 2000.

The papers were presented under the following headings:

- Academic Perspective: COTS Acquisition, Utilisation and Evaluation
- COTS Acquisition Challenges
- COTS: Evaluation and Assurance
- Vendor Perspective: COTS
- User Perspective: COTS
- COTS: Integration

NORTH ATLANTIC TREATY ORGANIZATION

RESEARCH AND TECHNOLOGY ORGANIZATION
BP 25 • 7 RUE ANCELLE
F-92201 NEUILLY-SUR-SEINE CEDEX • FRANCE
Télécopie 0(1)55.61.22.99 • E-mail mailbox@rta.nato.int

DIFFUSION DES PUBLICATIONS
RTO NON CLASSIFIEES

L'Organisation pour la recherche et la technologie de l'OTAN (RTO), détient un stock limité de certaines de ses publications récentes, ainsi que de celles de l'ancien AGARD (Groupe consultatif pour la recherche et les réalisations aérospatiales de l'OTAN). Celles-ci pourront éventuellement être obtenues sous forme de copie papier. Pour de plus amples renseignements concernant l'achat de ces ouvrages, adressez-vous par lettre ou par télécopie à l'adresse indiquée ci-dessus. Veuillez ne pas téléphoner.

Des exemplaires supplémentaires peuvent parfois être obtenus auprès des centres nationaux de distribution indiqués ci-dessous. Si vous souhaitez recevoir toutes les publications de la RTO, ou simplement celles qui concernent certains Panels, vous pouvez demander d'être inclus sur la liste d'envoi de l'un de ces centres.

Les publications de la RTO et de l'AGARD sont en vente auprès des agences de vente indiquées ci-dessous, sous forme de photocopie ou de microfiche. Certains originaux peuvent également être obtenus auprès de CASI.

CENTRES DE DIFFUSION NATIONAUX

ALLEMAGNE
Streitkräfteamt / Abteilung III
Fachinformationszentrum der Bundeswehr, (FIZBw)
Friedrich-Ebert-Allee 34
D-53113 Bonn

BELGIQUE
Coordinateur RTO - VSL/RTO
Etat-Major de la Force Aérienne
Quartier Reine Elisabeth
Rue d'Evère, B-1140 Bruxelles

CANADA
Directeur - Recherche et développement - Communications et gestion de l'information - DRDCGI 3
Ministère de la Défense nationale
Ottawa, Ontario K1A 0K2

DANEMARK
Danish Defence Research Establishment
Ryvangs Allé 1, P.O. Box 2715
DK-2100 Copenhagen Ø

ESPAGNE
INTA (RTO/AGARD Publications)
Carretera de Torrejón a Ajalvir, Pk.4
28850 Torrejón de Ardoz - Madrid

ETATS-UNIS
NASA Center for AeroSpace Information (CASI)
Parkway Center
7121 Standard Drive
Hanover, MD 21076-1320

FRANCE
O.N.E.R.A. (ISP)
29, Avenue de la Division Leclerc
BP 72, 92322 Châtillon Cedex

GRECE (Correspondant)
Hellenic Ministry of National Defence
Defence Industry Research & Technology General Directorate
Technological R&D Directorate
D.Soutsou 40, GR-11521, Athens

HONGRIE
Department for Scientific Analysis
Institute of Military Technology
Ministry of Defence
H-1525 Budapest P O Box 26

ISLANDE
Director of Aviation
c/o Flugrad
Reykjavik

ITALIE
Centro di Documentazione Tecnico-Scientifica della Difesa
Via XX Settembre 123a
00187 Roma

LUXEMBOURG
Voir Belgique

NORVEGE
Norwegian Defence Research Establishment
Attn: Biblioteket
P.O. Box 25, NO-2007 Kjeller

PAYS-BAS
NDRCC
DGM/DWOO
P.O. Box 20701
2500 ES Den Haag

POLOGNE
Chief of International Cooperation Division
Research & Development Department
218 Niepodleglosci Av.
00-911 Warsaw

PORTUGAL
Estado Maior da Força Aérea
SDFA - Centro de Documentação
Alfragide
P-2720 Amadora

REPUBLIQUE TCHEQUE
Distribuční a informační středisko R&T
VTÚL a PVO Praha
Mladoboleslavská ul.
197 06 Praha 9-Kbely AFB

ROYAUME-UNI
Defence Research Information Centre
Kentigern House
65 Brown Street
Glasgow G2 8EX

TURQUIE
Millî Savunma Başkanliği (MSB)
ARGE Dairesi Başkanliği (MSB)
06650 Bakanliklar - Ankara

AGENCES DE VENTE

NASA Center for AeroSpace Information (CASI)
Parkway Center
7121 Standard Drive
Hanover, MD 21076-1320
Etats-Unis

The British Library Document Supply Centre
Boston Spa, Wetherby
West Yorkshire LS23 7BQ
Royaume-Uni

Canada Institute for Scientific and Technical Information (CISTI)
National Research Council
Document Delivery
Montreal Road, Building M-55
Ottawa K1A 0S2, Canada

Les demandes de documents RTO ou AGARD doivent comporter la dénomination "RTO" ou "AGARD" selon le cas, suivie du numéro de série (par exemple AGARD-AG-315). Des informations analogues, telles que le titre et la date de publication sont souhaitables. Des références bibliographiques complètes ainsi que des résumés des publications RTO et AGARD figurent dans les journaux suivants:

Scientific and Technical Aerospace Reports (STAR)
STAR peut être consulté en ligne au localisateur de ressources uniformes (URL) suivant:
http://www.sti.nasa.gov/Pubs/star/Star.html
STAR est édité par CASI dans le cadre du programme NASA d'information scientifique et technique (STI)
STI Program Office, MS 157A
NASA Langley Research Center
Hampton, Virginia 23681-0001
Etats-Unis

Government Reports Announcements & Index (GRA&I)
publié par le National Technical Information Service
Springfield
Virginia 2216
Etats-Unis
(accessible également en mode interactif dans la base de données bibliographiques en ligne du NTIS, et sur CD-ROM)

Imprimé par St-Joseph Ottawa/Hull
(Membre de la Corporation St-Joseph)
45, boul. Sacré-Cœur, Hull (Québec), Canada J8X 1C6

NORTH ATLANTIC TREATY ORGANIZATION

RESEARCH AND TECHNOLOGY ORGANIZATION
BP 25 • 7 RUE ANCELLE
F-92201 NEUILLY-SUR-SEINE CEDEX • FRANCE
Telefax 0(1)55.61.22.99 • E-mail mailbox@rta.nato.int

DISTRIBUTION OF UNCLASSIFIED RTO PUBLICATIONS

NATO's Research and Technology Organization (RTO) holds limited quantities of some of its recent publications and those of the former AGARD (Advisory Group for Aerospace Research & Development of NATO), and these may be available for purchase in hard copy form. For more information, write or send a telefax to the address given above. **Please do not telephone**.

Further copies are sometimes available from the National Distribution Centres listed below. If you wish to receive all RTO publications, or just those relating to one or more specific RTO Panels, they may be willing to include you (or your organisation) in their distribution.

RTO and AGARD publications may be purchased from the Sales Agencies listed below, in photocopy or microfiche form. Original copies of some publications may be available from CASI.

NATIONAL DISTRIBUTION CENTRES

BELGIUM
Coordinateur RTO - VSL/RTO
Etat-Major de la Force Aérienne
Quartier Reine Elisabeth
Rue d'Evère, B-1140 Bruxelles

CANADA
Director Research & Development
Communications & Information
Management - DRDCIM 3
Dept of National Defence
Ottawa, Ontario K1A 0K2

CZECH REPUBLIC
Distribuční a informační středisko R&T
VTÚL a PVO Praha
Mladoboleslavská ul.
197 06 Praha 9-Kbely AFB

DENMARK
Danish Defence Research
Establishment
Ryvangs Allé 1, P.O. Box 2715
DK-2100 Copenhagen Ø

FRANCE
O.N.E.R.A. (ISP)
29 Avenue de la Division Leclerc
BP 72, 92322 Châtillon Cedex

GERMANY
Streitkräfteamt / Abteilung III
Fachinformationszentrum der
Bundeswehr, (FIZBw)
Friedrich-Ebert-Allee 34
D-53113 Bonn

GREECE (Point of Contact)
Hellenic Ministry of National
Defence
Defence Industry Research &
Technology General Directorate
Technological R&D Directorate
D.Soutsou 40, GR-11521, Athens

HUNGARY
Department for Scientific
Analysis
Institute of Military Technology
Ministry of Defence
H-1525 Budapest P O Box 26

ICELAND
Director of Aviation
c/o Flugrad
Reykjavik

ITALY
Centro di Documentazione
Tecnico-Scientifica della Difesa
Via XX Settembre 123a
00187 Roma

LUXEMBOURG
See Belgium

NETHERLANDS
NDRCC
DGM/DWOO
P.O. Box 20701
2500 ES Den Haag

NORWAY
Norwegian Defence Research
Establishment
Attn: Biblioteket
P.O. Box 25, NO-2007 Kjeller

POLAND
Chief of International Cooperation
Division
Research & Development
Department
218 Niepodleglosci Av.
00-911 Warsaw

PORTUGAL
Estado Maior da Força Aérea
SDFA - Centro de Documentação
Alfragide
P-2720 Amadora

SPAIN
INTA (RTO/AGARD Publications)
Carretera de Torrejón a Ajalvir, Pk.4
28850 Torrejón de Ardoz - Madrid

TURKEY
Millî Savunma Başkanliği (MSB)
ARGE Dairesi Başkanliği (MSB)
06650 Bakanliklar - Ankara

UNITED KINGDOM
Defence Research Information
Centre
Kentigern House
65 Brown Street
Glasgow G2 8EX

UNITED STATES
NASA Center for AeroSpace
Information (CASI)
Parkway Center
7121 Standard Drive
Hanover, MD 21076-1320

SALES AGENCIES

NASA Center for AeroSpace Information (CASI)
Parkway Center
7121 Standard Drive
Hanover, MD 21076-1320
United States

The British Library Document Supply Centre
Boston Spa, Wetherby
West Yorkshire LS23 7BQ
United Kingdom

Canada Institute for Scientific and Technical Information (CISTI)
National Research Council
Document Delivery
Montreal Road, Building M-55
Ottawa K1A 0S2, Canada

Requests for RTO or AGARD documents should include the word 'RTO' or 'AGARD', as appropriate, followed by the serial number (for example AGARD-AG-315). Collateral information such as title and publication date is desirable. Full bibliographical references and abstracts of RTO and AGARD publications are given in the following journals:

Scientific and Technical Aerospace Reports (STAR)
STAR is available on-line at the following uniform
resource locator:
http://www.sti.nasa.gov/Pubs/star/Star.html
STAR is published by CASI for the NASA Scientific
and Technical Information (STI) Program
STI Program Office, MS 157A
NASA Langley Research Center
Hampton, Virginia 23681-0001
United States

Government Reports Announcements & Index (GRA&I)
published by the National Technical Information Service
Springfield
Virginia 22161
United States
(also available online in the NTIS Bibliographic
Database or on CD-ROM)

Printed by St. Joseph Ottawa/Hull
(A St. Joseph Corporation Company)
45 Sacré-Cœur Blvd., Hull (Québec), Canada J8X 1C6

ISBN 92-837-1049-5